Communications and Networking

A Service-enabling Framework for the Session Initiation Protocol (SIP)

Gonzalo Camarillo

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 129/2011

A Service-enabling Framework for the Session Initiation Protocol (SIP)

Gonzalo Camarillo

Doctoral dissertation for the degree of Doctor of Science in Technology/ in Architecture (Doctor of Philosophy) to be presented with due permission of the School of Electrical Engineering for public examination and debate in Auditorium S3 at the Aalto University School of Electrical Engineering (Espoo, Finland) on the 13th of December 2011 at 12 noon (at 12 o'clock).

Aalto University
Aalto School of Electrical Engineering
Communications and Networking

Supervisor
Prof. Raimo Kantola (Aalto University, Finland)

Preliminary examiners
Prof. Sasu Tarkoma (Helsinki University, Finland)
Prof. Simon Pietro Romano (Universita degli Studi di Napoli Federico II, Italy)

Opponent
Prof. Arturo Azcorra (Universidad Carlos III de Madrid, Spain)

Aalto University publication series
DOCTORAL DISSERTATIONS 129/2011

ISBN 978-952-60-4402-6 (pdf)
ISBN 978-952-60-4401-9 (printed)
ISSN-L 1799-4934
ISSN 1799-4942 (pdf)
ISSN 1799-4934 (printed)

Unigrafia Oy
Helsinki 2011

Finland

The dissertation can be read at http://lib.tkk.fi/Diss/

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Gonzalo Camarillo

Name of the doctoral dissertation
A Service-enabling Framework for the Session Initiation Protocol (SIP)

Publisher Aalto School of Electrical Engineering

Unit Communications and Networking

Series Aalto University publication series DOCTORAL DISSERTATIONS 129/2011

Field of research Networking Technology

Manuscript submitted 9 November 2011 **Manuscript revised** 9 November 2011

Date of the defence 13 December 2011 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

In this dissertation, we propose a framework to provide multimedia communication services. Our proposed framework is based on SIP (Session Initiation Protocol) and has four fundamental properties: it is available, secure, high performing, and oriented to innovations. The framework is not an architecture with a rigid structure. Instead, the framework is a toolkit made up of a set of tools that can be combined in different ways. The combination of these tools provides applications and services with functionality needed to implement a wide variety of multimedia communication services. Applications and services built on top of the framework use different tools within the toolkit in order to provide their desired overall functionality.

The functionality provided by the framework includes a number of primitives to be used by applications and services. These primitives mostly relate to multiparty communications and include floor control. The framework also offers support functions that relate to PSTN (Public Switched Telephony Network) interworking, policy control, and consent-based communications. Additionally, the framework contains functions that relate to signalling transport, multihoming, mobility, security, and NAT (Network Address Translation) traversal. The framework also allows building overlay networks when a SIP network infrastructure is not available.

In order to test and refine the ideas presented in this dissertation, we have implemented most of them in proof-of-concept prototypes. We have used experiments and simulations to validate our assumptions and obtain new insights.

Keywords SIP, services, consent, SCTP, HIP, overlay

ISBN (printed) 978-952-60-4401-9 **ISBN (pdf)** 978-952-60-4402-6

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Espoo **Location of printing** Helsinki **Year** 2011

Pages 195 **The dissertation can be read at** http://lib.tkk.fi/Diss/

Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Gonzalo Camarillo

Väitöskirjan nimi
SIP-protokollaan pohjautuva viitekehys palvelujen mahdollistamiseen

Julkaisija Sähkötekniikan korkeakoulu

Yksikkö Tietoliikenne- ja tietoverkkotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 129/2011

Tutkimusala Tietoverkkotekniikka

Käsikirjoituksen pvm 09.11.2011 **Korjatun käsikirjoituksen pvm** 09.11.2011

Väitöspäivä 13.12.2011 **Kieli** Englanti

☐ **Monografia** ☒ **Yhdistelmäväitöskirja (yhteenveto-osa + erillisartikkelit)**

Tiivistelmä

Tässä väitöskirjassa ehdotamme viitekehystä multimediaviestintäpalvelujen tarjoamiseen. Ehdottamamme viitekehys pohjautuu SIP-protokollaan (Session Initiation Protocol) ja sillä on neljä merkittävää ominaisuutta: korkea käytettävyys, hyvä tietoturvan taso, korkea suorituskyky ja innovaatiosuuntautuneisuus. Viitekehysellämme ei ole tarkasti määriteltyä arkkitehtuuria. Tarkasti määritellyn arkkitehtuurin sijaan viitekehyksellä on joukko työkaluja, joita voidaan yhdistellä monin eri tavoin. Työkalujen erilaiset yhdistelmät tarjoavat sovelluksille ja palveluille ominaisuuksia, joita tarvitaan multimediaviestintäpalveluita toteutettaessa. Halutunlaisen kokonaistoiminnallisuuden saavuttamiseksi viitekehyksen päälle rakennetut sovellukset ja palvelut käyttävät erilaisia työkaluja.

Ehdottamamme viitekehyksen tarjoama kokonaistoiminnallisuus sisältää joukon alkeistoimintoja, joita sovellukset ja palvelut käyttävät. Nämä alkeistoiminnot liittyvät pääasiassa sellaisiin viestintätilanteisiin, joissa on useita osallistujia, ja ne sisältävät puheenvuorojen jakamiseen tarvittavaa tekniikkaa. Viitekehys tarjoaa myös sellaisia tukitoimintoja, jotka liittyvät yhteistoiminnan mahdollistamiseen perinteisen puhelinverkon kanssa, käytäntöjen hallintaan ja suostumuspohjaiseen viestintään. Tämän lisäksi viitekehys sisältää toimintoja, jotka liittyvät signaloinnin välittämiseen, useiden vaihtoehtoisten liityntäverkkojen käyttämiseen, liikkuvuuteen, tietoturvaan ja yhteyksien muodostamiseen osoitteenmuuttajien (NAT) lävitse. Viitekehys mahdollistaa myös vertaisverkkojen rakentamisen tilanteissa, joissa SIP-verkkoinfrastruktuuria ei ole käytettävissä.

Kehittääksemme ja testataksemme tässä väitöskirjassa esitettyjä ideoita olemme rakentaneet prototyyppejä useiden ideoiden pohjalta. Olemme tehneet kokeita ja simulaatioita vahvistaaksemme olettamuksiamme ja saadaksemme uusia ideoita jatkokehittelyä varten.

Avainsanat SIP, palvelut, suostumus, SCTP, HIP, vertaisverkko

ISBN (painettu) 978-952-60-4401-9 **ISBN (pdf)** 978-952-60-4402-6

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2011

Sivumäärä 195 **Luettavissa verkossa osoitteessa** http://lib.tkk.fi/Diss/

Preface

This dissertation includes the results of several years of research on the area of SIP-based services. I worked for Ericsson Finland during all those years. I would like to thank my colleagues and fellow researchers at Ericsson Finland for creating an environment that made it a pleasure to go to work every day. I would also like to thank my management for their support and encouragement during this period. In particular, the guidance provided by Leif Björklund, Roger Förström, Stefan von Schantz, and Rolf Svanbäck was essential when I decided to start my Ph.D. studies. Christian Engblom, Jussi Haapakangas, and Johan Torsner also supported me in my studies at later points in time. Raimo Vuopionpera never stopped encouraging me and showing genuine interest in how my research was progressing.

My supervisor at Aalto University (Helsinki University of Technology during most of my Ph.D. studies) was Professor Raimo Kantola. His advice during all my studies has been invaluable.

Pekka Nikander was my supervisor for some of the courses and seminars I took as part of my studies. He was also my colleague at Ericsson Finland for a number of years. Pekka's influence in my ideas and research has been substantial.

As part of my Ph.D. studies, I spent one and a half years at Columbia University (USA) as a visitor researcher at the laboratory led by Professor Henning Schulzrinne. Henning's ideas have been tremendously influential not only on my research but also on the whole area of real-time communications on the Internet.

I also want to thank the coauthors of the papers that make up this thesis. They were Henning Schulzrinne, Raimo Kantola, Tuomas Aura, Pekka Nikander, Tero Kauppinen, Martti Kuparinen, Ignacio Mas, Salvatore Loreto, Jani Hautakorpi, Ari Keränen, Sebastien Pierrel, Jouni

Maenpää, and Veera Andersson. I would also like to thank all the fellow researchers around the world with whom I have discussed and exchanged ideas during all these years.

The pre-examiners for this thesis were Professors Sasu Tarkoma and Simon Pietro Romano. I want to thank both of them for all their efforts reviewing it. Their comments and suggestions improved the quality of the thesis.

Lastly, I want to thank my family, girlfriend, and friends. They have been and continue to be a source of inspiration and motivation.

Helsinki, November 9, 2011,

Gonzalo Camarillo

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I G. Camarillo, T. Kauppinen, M. Kuparinen, and I. Más Ivar. Towards an Innovation Oriented IP Multimedia Subsystem. *IEEE Communications Magazine*, Vol. 45, No. 3, Pages 130-136, March 2007.

II G. Camarillo, Adding Consent-based Communications to SIP. Adding Consent-based Communications to SIP. *IEEE Vehicular Technology Magazine*, Vol. 2, No. 2, Pages 30-37, June 2007.

III G. Camarillo, H. Schulzrinne, and R. Kantola. Evaluation of Transport Protocols for the Session Initiation Protocol. *IEEE Network*, Vol. 17, No. 5, Pages 40-46, September 2003.

IV G. Camarillo, H. Schulzrinne, S. Loreto, and J. Hautakorpi. Effect of Head of the Line Blocking (HOLB) on Session Initiation Protocol (SIP) Session Establishment Delays. *Journal of Communications and Networks (JCN)*, Vol. 11, No. 1, Pages 72-83, February 2009.

V T. Aura, P. Nikander, and G. Camarillo. Effects of Mobility and Multihoming on Transport-Protocol Security. In *Proceedings of the IEEE Symposium on Security and Privacy (SSP)*, Pages 12-26, May 2004.

VI G. Camarillo, I. Más Ivars, and P. Nikander. A Framework to Combine the Session Initiation Protocol and the Host Identity Protocol. In *Proceedings of the IEEE Wireless Communications and Networking Con-*

ference (WCNC), Pages 3051-3056, May 2008.

VII G. Camarillo, A. Keränen, and S. Pierrel. Automatic Flow-specific Multi-path Management for the Host Identity Protocol (HIP). In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Pages 1-6, April 2010.

VIII G. Camarillo, J. Mäenpää, A. Keränen, and V. Andersson. Reducing Delays Related to NAT Traversal in P2PSIP Session Establishments. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC)*, Pages 549-553, January 2011.

Author's Contribution

Publication I: "Towards an Innovation Oriented IP Multimedia Subsystem"

The author of this dissertation was the main author of this paper. His contribution consisted of providing the main idea for the paper and acting as the main editor of the paper.

Publication II: "Adding Consent-based Communications to SIP"

The author of this dissertation was the sole author of this paper.

Publication III: "Evaluation of Transport Protocols for the Session Initiation Protocol"

The author of this dissertation was the main author of this paper. His contribution consisted of providing the main idea for the paper, implementing the prototype, performing the experiments, measuring the results, performing the statistical analysis, and acting as the main editor of the paper.

Publication IV: "Effect of Head of the Line Blocking (HOLB) on Session Initiation Protocol (SIP) Session Establishment Delays"

The author of this dissertation was the main author of this paper. His contribution consisted of providing the main idea for the paper, providing the configurations for the experiments, performing the statistical analy-

sis, and acting as the main editor of the paper.

Publication V: "Effects of Mobility and Multihoming on Transport-Protocol Security"

The author of this dissertation was one of the coauthors of this paper. His contribution consisted of discussing and improving the ideas in the paper by having brainstorming sessions with the other coauthors, finding attacks and countermeasures in the context of telephony signalling and real-time media transmissions, studying the behavior of firewalls in different circumstances, and editing parts of the paper.

Publication VI: "A Framework to Combine the Session Initiation Protocol and the Host Identity Protocol"

The author of this dissertation was the main author of this paper. His contribution consisted of providing the main idea for the paper, providing the configurations for the experiments, analyzing the results, and acting as the main editor of the paper.

Publication VII: "Automatic Flow-specific Multi-path Management for the Host Identity Protocol (HIP)"

The author of this dissertation was the main author of this paper. His contribution consisted of providing the main idea for the paper and acting as the main editor of the paper.

Publication VIII: "Reducing Delays Related to NAT Traversal in P2PSIP Session Establishments"

The author of this dissertation was the main author of this paper. His contribution consisted of providing the main idea for the paper and acting as the main editor of the paper.

List of Abbreviations

3GPP	3rd Generation Partnership Project
AF	Application Function
AoR	Address of Record
API	Application Programming Interface
B2BUA	Back-to-Back User Agent
BEET	Bound End to End Tunnel
BEX	Base Exchange
BFCP	Binary Floor Control Protocol
CGA	Cryptographically Generated Address
CGI	Common Gateway Interface
CSCF	Call State Control Function
CWND	Congestion Window
DHT	Distributed Hash Table
DNS	Domain Name System
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
ECN	Explicit Congestion Notification
ESP	Encapsulating Security Payload
FBC	Flow Based Charging
FIFO	First In First Out

FQDN	Fully Qualified Domain Name
GPRS	General Packet Radio Service
HIP	Host Identity Protocol
HIP BONE	Host Identity Protocol Based Overlay Networking Environment
HIT	Host Identity Tag
HOL	Head of Line
HOLB	Head of the Line Blocking
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICE	Interactive Connectivity Establishment
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IMS	Internet Protocol Multimedia Subsystem
IP	Internet Protocol
IPsec	Internet Protocol Security
IPTV	Internet Protocol Television
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU	International Communication Union
JSR	Java Specification Request
MAC	Message Authentication Code
MAT	Mail Transfer Agent
MIME	Multipurpose Internet Mail Extensions
MIP	Mobile Internet Protocol
MTU	Maximum Transmission Unit
NAT	Network Address Translation

NNI	Network to Network Interface
OMA	Open Mobile Alliance
ORCHID	Overlay Cryptographic Hash Identifiers
P-CSCF	Proxy Call State Control Function
P2P	Peer to Peer
P2PP	Peer to Peer Protocol
P2PSIP	Peer-to-peer Session Initiation Protocol
PBX	Private Branch Exchange
PCC	Policy and Charging Control
PCM	Pulse Code Modulation
PCRF	Policy and Charging Rules Function
PDA	Personal Digital Assistant
PoC	Push to talk over Cellular
PSTN	Public Switched Telephony Network
QoS	Quality of Service
RELOAD	Resource Location And Discovery
RFC	Request for Comments
RLS	Resource List Service
ROHC	Robust Header Compression
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTT	Roundtrip Time
S-CSCF	Serving Call State Control Function
SBC	Session Border Controller
SBLP	Service Based Local Policy
SCTP	Stream Control Transmission Protocol

SDP	Session Description Protocol
SEND	Secure Neighbor Discovery
SIMA	Simultaneous Multiaccess
SIP	Session Initiation Protocol
SIP-I	Session Initiation Protocol with encapsulated ISUP
SIP-T	Session Initiation Protocol for Telephones
SIPS	Session Initiation Protocol Secure
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPIT	Spam over Internet Telephony
SRF	Single Reservation Flow
STUN	Simple Traversal of User Datagram Protocol
TCB	Transmission Control Block
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TURN	Traversal Using Relays around NAT
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UNI	User to Network Interface
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
XCAP	Extensible Markup Language Configuration Access Protocol
XML	Extensible Markup Language

1. Introduction

The tremendous success of the Internet as a platform for all kinds of innovations and services is having a profound impact in many industries. Whole industries are looking into using the Internet to provide their services, which were previously provided through service-specific infrastructures.

Two examples of services migrating from their own infrastructures to the Internet are voice communications and television. Voice communication services previously provided over the PSTN (Public Switched Telephony Network) are becoming VoIP (Voice over Internet Protocol) services while television services previously provided over the broadcast television network are becoming IPTV services. Service providers offering Internet access, VoIP, and IPTV are said to provide a triple play service. This type of convergence is seen as a means to provide innovative services in a cost-effective manner.

However, service migration is not about simply replicating the legacy service using Internet technologies. It is about providing a more appealing service taking advantage of state-of-the-art Internet technologies. It represents a unique chance to rethink whole services (and whole industries), keep their good features, and add new ones.

This thesis is about multimedia communication services. The idea was to build a framework to provide these types of services on the Internet. We chose to base our framework on SIP (Session Initiation Protocol) [124], which is a protocol intended to establish and manage multimedia sessions. Before designing our framework, we gathered a set of requirements in the form of features our framework had to support. These requirements guided the whole design of the framework. As discussed in Chapter 2, the features we chose relate to availability, security, performance, and innovativeness.

This thesis is a bundle of scientific papers, each of which deals with a different part of the framework. This summary puts these papers into context and describes the framework that links them together.

1.1 Research Methodology

The research methodology we have used in this thesis aimed to produce original scientific results that were relevant and as widely used as possible by the Internet community. In the process whereby our initial ideas became what is documented in the scientific papers that make up this thesis, we made extensive use of proof-of-concept prototypes and test beds. Implementing our ideas helped us refine them and make better decisions, which were based on running code and preliminary measurements.

As we stated earlier, one of our main goals was to produce results that were relevant and as widely used as possible. We like to measure our results not only by their scientific value in terms of originality, innovativeness, elegance, etc., but also in terms of their actual impact (i.e., their use in deployments on the Internet). In order to make a significant impact, it is not enough to resolve research problems at a high level. In addition to its associated research issues, a considerable number of engineering issues need to be resolved for an idea to be implemented beyond proof-of-concept prototypes. Additionally, the standardization of the results is key to enable multiple interoperable implementations.

Consequently, in addition to working on the scientific papers that form the core of this thesis, we put considerable efforts on resolving and standardizing engineering issues related to them. As a result, we have produced a comprehensive framework that covers aspects ranging from high-level research issues to small implementation details that are nevertheless essential to enable interoperability. While our research work is the core of this framework, our engineering work is the glue that links all our research ideas in different areas to form a coherent framework. For example, in order to study the effects different transport protocols have on SIP, we had to specify how those transport protocols could carry SIP. Likewise, in order to add consent-based communications to SIP, we had to specify a format to write permission documents. While we describe both our research and our engineering work in this thesis, we obviously discuss our research work more thoroughly.

1.2 Structure of the Thesis

This thesis describes a service-enabling framework based on SIP. As stated earlier, even though, officially, this thesis is a bundle of scientific papers, the body of work supporting it is considerably larger. We have authored or coauthored a considerable number of RFCs (Requests for Comments) that are directly related to the papers that form this thesis. The chapters describing the different parts of the framework discuss both our scientific results (published in 8 scientific papers) and our engineering work (published in 25 RFCs[1], which are referenced below) in the area. They also discuss related work in the area. The remainder of this thesis is organized as follows.

Chapter 2 describes the goal of this thesis, which consisted of creating a framework that met a set of requirements. The section also discusses those requirements.

Chapter 3 provides readers with background on SIP. Given that the framework discussed in this thesis is based on SIP, knowing what functionality SIP provides and how it provides it is essential for the understanding of the thesis. The section is based on our work on the main SIP specification, which is documented in [124], and on our two SIP-related books: [11] and [18].

Chapter 4 provides an overview of the framework. It describes the whole framework and its individual components at a high level.

Chapter 5 discusses the main primitives we added to our framework. The section is based on our work on the main SIP specification, which is documented in [124], our work on URI-list services, which is documented in [29], [43], [44], [21], [30], and [24], and our work on BFCP (Binary Floor Control Protocol), which is documented in [26], [13], and [14].

Chapter 6 discusses the support functions we added to our framework. The section is based on our work on PSTN interworking, which is documented in [27], and [28], our work on session policies, which is documented in Publication I, [57], [59], [58], and [61], and our work on consent-based communications, which is documented in Publication II, [117], [118], [15], and [16].

Chapter 7 discusses the transport properties of our framework. The section is based on our work on transporting SIP, which is documented in

[1] at the time of writing, four of these documents are still in the Internet Draft state and will be published as RFCs shortly.

Publication III, Publication IV, and [123].

Chapter 8 discusses the multihoming, mobility, security, and NAT (Network Address Translation) traversal properties of our framework. The section is based on our work on increasing SCTP's (Stream Control Transmission Protocol) security, which is documented in Publication V, and [135], our work on combining HIP (Host Identity Protocol) and SIP, which is documented in Publication VI, and our work on traffic-flow management in HIP, which is documented in Publication VII.

Chapter 9 discusses how to apply our framework to scenarios without infrastructure elements. The section is based on our work on building HIP-based overlays, which is documented in Publication VI, Publication VIII, [25], [73], [23], and [22].

Chapter 10 contains the conclusions of this thesis. In this section, we analyze how our framework meets its original requirements.

2. Research Goal

Our goal was to build a service-enabling framework for multimedia communications that supported a set of fundamental properties. We used these properties as design requirements to build the framework. Our framework needed to be available, secure, high performing, and oriented to innovations. We discuss why we chose these properties and our main design decisions in the following sections.

2.1 Framework

The terms “framework” and “architecture” mean several different things in the existing literature. These terms are even often used without being defined. One of the reasons for the lack of a single definition is the different levels of abstractions at which frameworks and architectures can be defined (e.g., software framework and protocol framework). In this dissertation, we define these terms as follows.

A framework provides a generic skeleton for combining functional elements to implement applications or services. The architecture of a particular application or service describes the way elements of the framework are combined and used to build the application or service.

We aimed to develop a protocol framework whose elements were common primitives and protocols providing generic functionality. An intuitive way to think about a framework and its elements is as a toolkit and the tools inside it. Thus, the elements in our framework can be seen as the tools in a toolkit, which can be combined in different ways to implement a wide variety of multimedia communication services. The productivity of application and service developers is improved by allowing them to focus on the unique requirements of their applications and services instead of

developing generic functionality, which is available within the framework.

2.2 Multimedia Communications based on SIP

The first decision we needed to make was to choose a signalling protocol for our framework. We wanted a standard protocol with an open specification so that it could be implemented and deployed by the Internet community as a whole. At the time of making our decision, the two main standard signalling protocols to manage multimedia sessions on the Internet were SIP [124] and H.323 [67]. We chose to base our framework on SIP. One of the main reasons why we chose SIP was that, at that point, SIP was in its infancy and there was still plenty of room to introduce some of our ideas in the protocol as it evolved.

Nowadays, SIP is the most widely-used standard protocol to establish voice and video sessions on the Internet. It is also widely used in enterprise networks and telecom environments. However, at present, there are still a set of closed systems that use proprietary protocols. Nevertheless, some of the most widely-used closed VoIP systems, such as Skype [132], also use SIP to interconnect with the PSTN and with SIP-based enterprise networks.

Our goal when working on specifying SIP was to make it a signalling protocol able to support a communications framework with the features we had chosen. We discuss these features in the following sections.

2.3 Service-enabling

Our framework aimed to enable a wide range of applications to provide communication services. A service-enabling framework needs to provide a complete set of primitives, which should include the most common functionality required by the services to be developed on top of the framework. SIP natively provides most of the primitives commonly used by applications handling one-to-one sessions (e.g., establish session to a given URI, modify session, and terminate session). Additionally, SIP can easily manage sessions whose media is multicast. However, vanilla SIP does not provide simple primitives to handle certain types of scenarios involving multiparty communications in an efficient way (e.g., setting up a confer-

ence session and inviting a large number of users to it). Additionally, there were no mechanisms to handle sessions that included some form of floor control (i.e., the ability to manage the access to a shared resource).

We consider the management of multiparty sessions as a fundamental property of our framework. Therefore, we took the ability to efficiently manage multiparty sessions with and without floor control as one of our requirements.

Note that when we use the term primitives we refer to protocol primitives. Applications use these protocol primitives to perform different functions. In practice, some applications do not access the protocol primitives provided by the framework directly. Instead, they access the protocol primitives through different APIs (e.g., JSR 32 [105], JSR 141 [104], JSR 180 [108], JSR 281 [107], JSR 289 [106], or Parlay X [3]). These APIs are outside the scope of this thesis.

2.4 Fundamental Properties

We chose four fundamental properties and used them as design requirements to build the framework. We decided our framework needed to be available, secure, high performing, and oriented to innovations. We considered these properties to be essential for our framework to be useful in the current business and technological Internet environment.

The choice of the first three properties is fairly natural. Security, performance, and availability are generally considered the main properties in VoIP deployments [99]. They are also among the most important quality criteria for success of web applications [94] and the main quality attributes in software architectures [70]. Additionally, security, performance, and availability are the three top customer requirements for public cloud environments [45].

As stated previously, our framework provides elements that can be combined in different ways to implement a wide variety of multimedia communication services. A given individual deployment chooses and combines a set of those elements in a particular way to provide the required functionality. Consequently, the goal is for our framework to include functionality that can be used by individual deployments to achieve those fundamental properties. The degree in which an individual deployment will posses any of the fundamental properties will depend on its choices.

Therefore, while our service-enabling framework includes building blocks to achieve these three properties, particular services or applications based on our framework will not typically be able to maximize all the properties at the same time. Consequently, different deployments will implement different compromises. For example, there exists a trade off between security and performance [35]. Encrypting and decrypting traffic increases the security of a system but decreases its performance. Compromises will be implemented based on the requirements each particular deployment needs to meet.

Users of traditional communication networks such as the PSTN are used to the first three of our properties (i.e., availability, security, and high performance). New communication networks based on disruptive technologies (i.e., Internet technology) aiming to attract a high number of users need to have acceptable (i.e., good-enough) levels in all three areas [32].

However, while having these three properties is a necessary condition for a communication network to be successful, such a network needs to provide something else. As the success of the Internet as a service platform has shown, users appreciate new and innovative services. The more appealing the services a communication network provides, the more users it will attract. Therefore, our framework needs to enable service developers to continuously create innovative services [32]. Consequently, we added this fourth property (i.e., innovativeness) to our list of fundamental properties.

2.4.1 Availability

We wanted our framework to be available in several ways. Our framework needed to support mobility in order to avoid service disruptions even when users are on the move. The framework needs to support multihoming so that services are not disrupted when a given access becomes unavailable. Availability under DoS (Denial of Service) attacks was also an important feature we wanted. An essential requirement for any framework to be used on the public Internet is that it can be used in the presence of NATs. Additionally, we wanted our framework to be available even in cases where there was no available infrastructure (e.g., ad-hoc environments or disaster areas). To make our framework available to a larger number of users, it had to be possible to access it from the legacy tele-

phone network (i.e., the PSTN). Of course, PSTN users may be able to access the framework in a limited way, but interworking with the PSTN is currently essential for any network providing communication services.

2.4.2 Security

The confidentiality and integrity of both signalling and media traffic in our network was a key requirement. Although there are several mechanisms available in this area, we wanted mechanisms that were as efficient as possible. Therefore, their integration with our framework had to be done in a way that minimizes establishment time and management complexity.

2.4.3 High Performance

Performance is a key issue for communication services. Users are fairly sensitive to delays (and to the variation of the delays) in the establishment of their sessions. We took the efficient transport of signalling messages between the different nodes in our framework as a requirement. Additionally, when using SIP in some scenarios, some operations are performed several times (e.g., connectivity establishment in the presence of NATs). We aimed to increase the performance of our framework by removing unnecessary duplications in our operations.

2.4.4 Innovativeness

Our framework aimed to allow applications running on top of it to be as innovative as possible. Applications using our framework do not need to be unnecessarily restricted by the framework itself in what type of new functionality they want to provide. For example, if an application needs to establish a session that includes a media stream of a new media type (e.g., smell or hologram), it should be able to do so even if that media type was not known when our framework was developed or deployed. Our framework needs to be future proof so that new innovative communication services can be provided on top of it.

2.5 Contribution to Knowledge

The following are the main contributions to knowledge of this dissertation:

- A mechanism that allows network intermediaries to apply session policies without breaking the end-to-end properties of session negotiations between SIP user agents. The mechanism enables the introduction of new and innovative services at a much faster rate because intermediaries do not need to be updated every time a new service is to be provided.

- A mechanism for implementing consent-based communications in SIP. The mechanism addresses the unwanted traffic issue by keeping users from receiving communication attempts from unauthorized users.

- A thorough study of the impact of HOLB (Head of the Line Blocking) on the transport of SIP messages. Under network conditions where the effects of HOLB become significant, using SCTP instead of TCP reduces session establishment delays substantially.

- A thorough security study of the dynamic addressing features in SCTP. We identified vulnerabilities in how those features were implemented and proposed ways to fix them. As a result, the relevant SCTP specifications were modified in order to address those vulnerabilities.

- A framework and a set of mechanisms for building HIP-based overlay networks. Having HIP manage all the data connections between two nodes in an integrated manner results in a significant reduction in session establishment delays in the presence of NATs

3. The Session Initiation Protocol (SIP)

Even though SIP [124] was initially designed to invite users to existing multimedia conferences, today it is mainly used to create, modify and terminate multimedia sessions. In addition, there exist SIP extensions to deliver instant messages and to handle subscriptions to events. In this thesis, we focus on the most widespread use of SIP: the session establishment and management functionality it provides.

3.1 SIP Functionality

The main goal of SIP is to deliver a session description to a user at his or her current location. Once the user has been located and the initial session description has been delivered, SIP can deliver new session descriptions to modify the characteristics of the ongoing sessions or terminate the session at the user's request.

3.1.1 Session Descriptions and SDP

A session description is, as its name indicates, a description of the session to be established. It contains enough information for the remote user to join the session. In multimedia sessions over the Internet this information includes the IP address and port number where the media needs to be sent, and the codecs used to encode voice and video.

The most common format for describing multimedia sessions is SDP (Session Description Protocol) [52]. Note that although the "P" in SDP stands for "Protocol", SDP is simply a textual format for describing multimedia sessions. Figure 3.1 shows an example of an SDP session description sent by the user agent of a user called "Alice". It contains, among other things, the subject of the conversation (swimming techniques), the

user agent's IP address (192.0.2.1), the port number where the user agent wants to receive audio (20000), the port number where the user agent wants to receive video (20002), and the audio and video codecs the user agent supports (0 corresponds to the audio codec G.711 μ-law and 31 corresponds to the video codec H.261).

```
v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.2.1
s=Let's talk about swimming techniques
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=sendrecv
m=video 20002 RTP/AVP 31
a=sendrecv
```

Figure 3.1. Example of an SDP session description

An SDP description consists of two parts: session-level information and media-level information. The session-level information applies to the whole session and comes before the `m=` lines. In our example, the first five lines correspond to session-level information. They provide version and user identifiers (`v=` and `o=` lines), the subject of the session (`s=` line), Alice's IP address (`c=` line), and the time of the session (`t=` line). Note that this session is supposed to take place at the moment when this session description is received. That is why the `t=` line is `t=0 0`.

The media-level information is media-stream specific and consists of an `m=` line and a number of optional `a=` lines that provide further information about the media stream. Our example has two media streams and, thus, has two `m=` lines. The `a=` lines indicate that the streams are bidirectional (i.e., users send and receive media).

As Figure 3.1 illustrates, the format of all the SDP lines consists of *type=value* elements, where type is always one character long. Table 3.1 shows the types defined by SDP.

Even if SDP is the most common format to describe multimedia sessions, SIP does not depend on it. SIP is session-description format independent. That is, SIP can deliver a description of a session written in SDP or in any other format.

In general, SIP is completely independent of the format of the objects it transports. Those objects may be session descriptions written in different

Table 3.1. SDP types

Type	Meaning
v	Protocol version
b	Bandwidth information
o	Owner of the session and session identifier
z	Time zone adjustments
s	Name of the session
k	Encryption key
i	Information about the session
a	Attribute lines
u	URL containing a description of the session
t	Time when the session is active
e	Email address to obtain information about the session
t	Times when the session will be repeated
p	Phone number to obtain information about the session
m	Media line
c	Connection information
i	Information about the media line

formats or any other piece of information. For example, there is an extension to SIP that allows SIP to deliver instant messages, which are written using a different format from SDP (e.g., html).

3.1.2 The Offer/Answer Model

SIP provides a two-way session-description exchange called the offer/answer model [121]. One of the users (the offerer) generates a session description (the offer) and sends it to the remote user (the answerer), who then generates a new session description (the answer) and sends it to the offerer.

After the offer/answer exchange, both users have a common view of the session to be established. They know, at least, the formats they can use (i.e., formats that the remote end understands) and the transport addresses for the session. The offer/answer exchange can also provide extra information, such as cryptographic keys for encrypting traffic.

Figure 3.2 shows the answer that the user agent of the user called “Bob” returned after having received the offer in Figure 3.1. The user agent’s IP address is 192.0.2.2, the port number where it will receive audio is 30000,

the port number where it will receive video is 30002. In this answerer, the user agent supports the same audio and video codecs as offerer (G.711 μ-law and H.261). So, after this offer/answer exchange, the session is established.

```
v=0
o=Bob 234562566 236376607 IN IP4 192.0.2.2
s=Let's talk about swimming techniques
c=IN IP4 192.0.2.2
t=0 0
m=audio 30000 RTP/AVP 0
a=sendrecv
m=video 30002 RTP/AVP 31
a=sendrecv
```

Figure 3.2. An SDP answer

3.1.3 SIP and SIPS URIs

SIP identifies users by their SIP URIs, which consist of a username and a domain name. In addition, SIP URIs can contain a number of parameters (e.g., transport), which are encoded using semicolons. The following are examples of SIP URIs:

```
sip:Alice.Smith@domain.com
sip:Bob.Brown@example.com
sip:carol@ws1234.domain2.com;transport=tcp
```

In addition, users can be identified using SIPS URIs. Entities contacting a SIPS URI use TLS (Transport Layer Security) [37] to secure their messages. The following are examples of SIPS URIs:

```
sips:Alice.Smith@domain.com
sips:Bob.Brown@example.com
```

3.1.4 User Location

SIP provides personal mobility. That is, users can be reached using the same identifier no matter where they are. For example, the user Alice can be reached at

```
sip:Alice.Smith@domain.com
```

regardless of her current location. This is her public URI, also known as her AoR (Address of Record).

Nevertheless, when Alice is logged in at work her SIP URI is

```
sip:asmith@ws1234.company.com
```

and when she is working at her computer at the university her SIP URI is

```
sip:alice@pc12.university.edu
```

SIP provides a means for mapping Alice's public URI

```
sip:Alice.Smith@domain.com
```

to her current URI (at work or at the university) at any given moment.

To do this, SIP introduces a network element called the registrar of a particular domain. A registrar handles requests addressed to its domain. Thus, SIP requests sent to

```
sip:Alice.Smith@domain.com
```

will be handled by the SIP registrar at domain.com.

Every time Alice logs into a new location, she registers her new location with the registrar at domain.com, as shown in Figure 3.3. In this way, the registrar at domain.com can always forward incoming requests to Alice wherever she is.

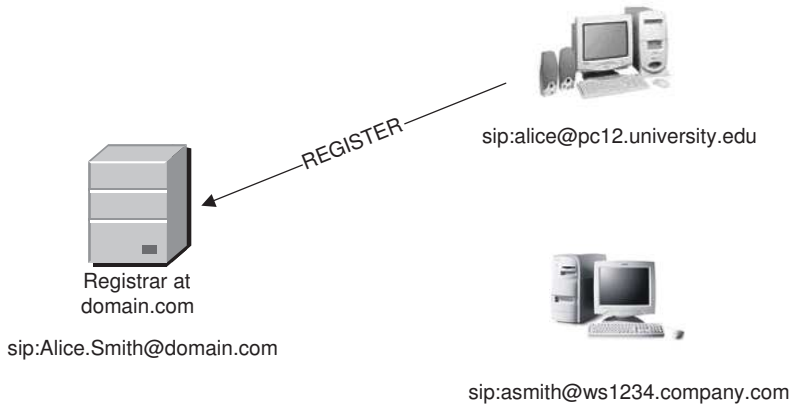

Figure 3.3. Alice registers her location with the domain.com registrar

On reception of the registration the registrar at domain.com can store the mapping between Alice's public URI and her current location in two

ways: it can use a local database or it can upload this mapping into a location server. If the registrar uses a location server, it will need to consult it when it receives a request for Alice. Note that the interface between the registrar and the location server is not based on SIP, but on other protocols.

3.2 SIP Entities

In addition to registrars, SIP defines user agents, proxy servers, and redirect servers. UAs (user agents) are SIP endpoints that are usually handled by a user, although user agents can also establish sessions automatically with no user intervention (e.g., a SIP voicemail). Sessions are typically established between user agents. Within a request/response SIP transaction, the user agent sending the request is referred to as the UAC (User Agent Client). The user agent sending the response is referred to as the UAS (User Agent Server). During a session, a user agent may act as the UAC in some transactions and as the UAS in others.

Proxy servers, typically referred to as proxies, are SIP routers. A proxy receives a SIP message from a user agent or from another proxy and routes it towards its destination. Routing the request involves relaying the message to the destination user agent or to another proxy on the path.

SIP routing is one of the key components of the protocol. A given user can be available at several user agents at the same time. For instance, Alice can be reachable on her computer at the university

```
sip:alice@pc12.university.edu
```

and on her PDA with a wireless connection

```
sip:alice@pda.com
```

She has registered both locations with the registrar at domain.com. If the registrar receives a SIP message addressed to Alice's public URI

```
sip:Alice.Smith@domain.com
```

it has to decide whether to route it to Alice's computer or to Alice's PDA. In this case, Alice has programmed the registrar to route SIP messages to her computer between 8:00 and 13:00 and to her PDA from 13:00 to 14:00. The registrar simply checks the current time and routes the SIP message accordingly.

Being able to route SIP messages on the basis of any criteria is a very powerful tool for building services that are specially tailored to the needs of each user. Users typically choose to route SIP messages based on the sender, the time of the day, whether the subject is business-related or personal, the type of session (e.g., route video calls to the computer with the big screen), etc.; the combinations are numerous.

In the previous example we saw that the registrar routed the SIP message to Alice's user agent. Yet the entities handling routing of messages are called proxies. Proxies and registrars are only logical roles. In our example, the same physical box acted as a registrar when Alice registered her current location and as a proxy when it was routing SIP messages toward Alice's user agent. This configuration is shown in Figure 3.4.

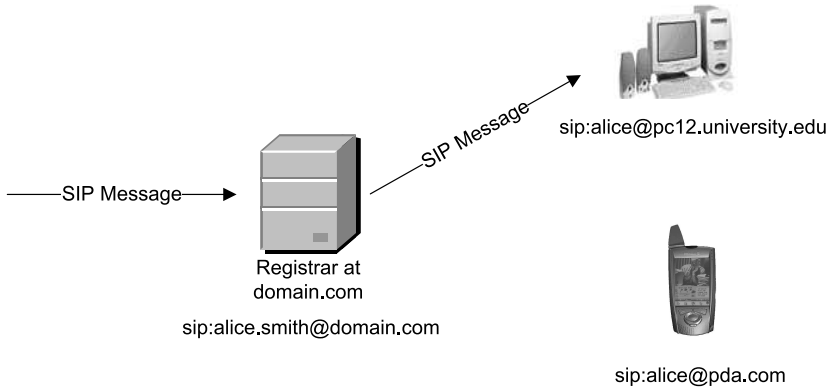

Figure 3.4. Proxy co-located with the registrar of the domain

A different configuration could consist of using a separate physical box for each role, as shown in Figure 3.5. Here, the proxy needs to access the information about Alice's location that the registrar got in the first place. This is resolved by adding a location server. The registrar uploads Alice's location to the location server, and the proxy consults the location server in order to route incoming messages.

3.2.1 Forking Proxies

In the previous examples the proxy chose a single user agent as the destination of the SIP message. However, sometimes it is useful to receive calls on several user agents at the same time. SIP proxy servers that route messages to more than one destination are called forking proxies, as shown in Figure 3.6.

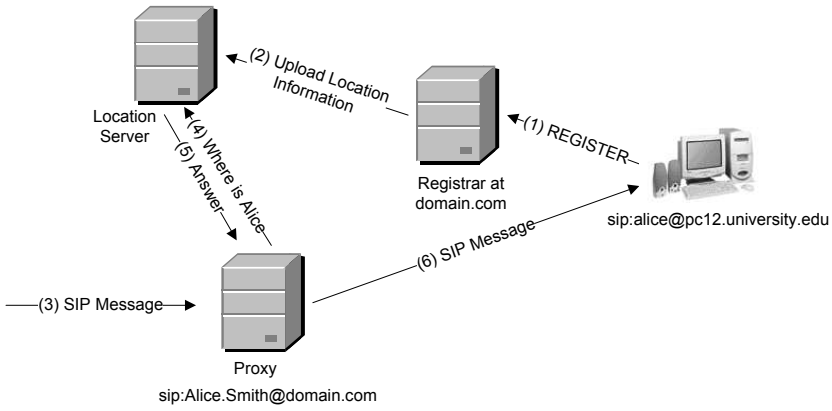

Figure 3.5. Proxy and registrar kept separate

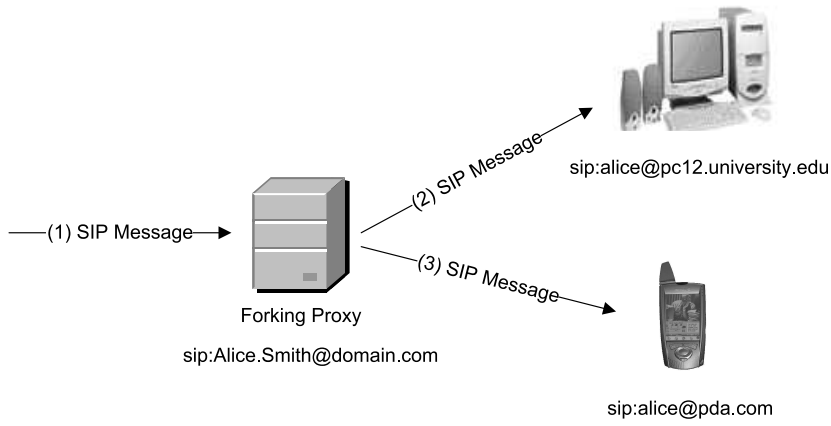

Figure 3.6. Forking proxy operation

A forking proxy can route messages in parallel or in sequence. An example of parallel forking is the simultaneous ringing of all the telephones in a house. Sequential forking consists of the proxy trying the different locations one after the other. A proxy can, for example, let a user agent ring for a certain period of time and, if the user does not pick up, try a new user agent.

Proxies store different amounts of information depending on the service they provide. Proxies can be stateless, (transaction) stateful, and call stateful. Stateless proxies do not store any state information after having routed a message. They are typically used as load balancers and entry points to proxy farms.

Stateful or transaction stateful proxies store state for ongoing transactions. For example, a forking proxy trying different locations sequentially could be implemented with a transaction stateful proxy. Such a proxy would store a list of locations to be tried and would remove them from the list as they were tried in sequence.

Call stateful proxies store state for ongoing sessions. A proxy that logged the duration of a session could be implemented with a call stateful proxy. Such a proxy would start a timer on session establishment and stop it on session termination.

3.2.2 Redirect Servers

Redirect servers are also used to route SIP messages, but they do not relay the message to its destination as proxies do. Redirect servers instruct the entity that sent the message (a user agent or a proxy) to try a new location instead. Figure 3.7 shows how redirect servers work. A user agent sends a SIP message to

`sip:Alice.Smith@domain.com`

and the redirect server tells it to try the alternative address

`sip:alice@pda.com`

3.3 Message Format

SIP is based on HTTP [41] and so it is a textual request-response protocol. Clients send requests, and servers answer with responses. A SIP transac-

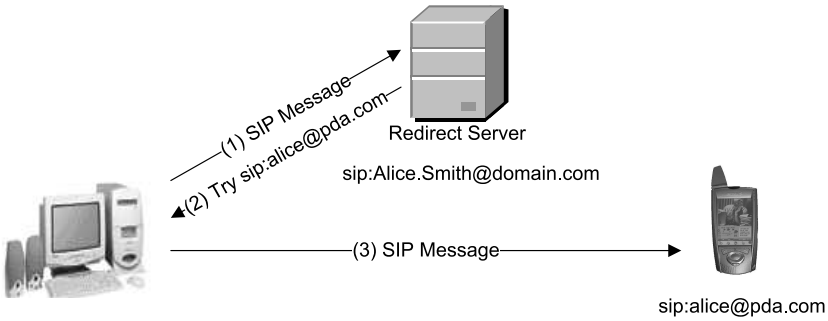

Figure 3.7. Redirect server operation

tion consists of a request from a client, zero or more provisional responses, and a final response from a server.

Figure 3.8 shows the format of SIP messages. They start with the *start line*, which is called the *request line* in requests and the *status line* in responses. The *start line* is followed by a number of header fields that follow the format *name:value* and an empty line that separates the header fields from the optional message body.

```
Start line
A number of header fields
Empty line
Optional message body
```

Figure 3.8. SIP message format

3.4 The Start Line in SIP Requests: the Request Line

The start line in requests is referred to as the request line. It consists of a *method name*, the *Request-URI*, and the protocol version `SIP/2.0`. The method name indicates the purpose of the request and the *Request-URI* contains the destination of the request. The following is an example of a request line:

```
INVITE sip:Alice.Smith@domain.com SIP/2.0
```

The method name in this example is `INVITE`. It indicates that the purpose of this request is to invite a user to a session. The *Request-URI* shows that this request is intended for Alice.

Table 3.2 shows the methods that are currently defined in SIP and their meaning.

Table 3.2. SIP methods

Method name	Meaning
ACK [124]	Acknowledges a final response for an INVITE
BYE [124]	Terminates a session
CANCEL [124]	Cancels a pending request
INFO [39]	Transports PSTN telephony signaling
INVITE [124]	Establishes a session
NOTIFY [111]	Notifies the user agent about a particular event
OPTIONS [124]	Queries a server about its capabilities
PRACK [122]	Acknowledges the reception of a provisional response
PUBLISH [88]	Uploads information to a server
REGISTER [124]	Maps a public URI with the current location of the user
SUBSCRIBE [111]	Requests to be notified about a particular event
UPDATE [113]	Modifies some characteristics of a session
MESSAGE [31]	Carries an instant message
REFER [133]	Instructs a server to send a request

Figure 3.9 shows a SIP transaction. The user agent client (UAC) sends a BYE request, and the user agent server (UAS) sends back a 200 (OK) response. Note that, usually, SIP message flows only show the method name of the request and the status code and the reason phrase of the response. These pieces of information are usually enough for any message flow to be understood.

3.5 The Start Line in SIP Responses: the Status Line

The start line of a response is referred to as the status line. The status line contains the protocol version (SIP/2.0) and the status of the transaction, which is given in numerical (status code) and human-readable (reason phrase) formats. The following is an example of a status line:

```
SIP/2.0 180 Ringing
```

The protocol version is always set to `SIP/2.0`. SIP can be extended without the need to increase its protocol version.

The status code `180` indicates that the remote user is being alerted. `Ringing` is the reason phrase and it is intended to be read by a human (e.g., displayed to the user). Since it is intended for human consumption the reason phrase can be written in any language.

Responses are classified by their status codes, which are integers that range from 100 to 699. Table 3.3 shows how status codes are classified according to their values.

Table 3.3. Status code ranges

Status code range	Meaning
100–199	Provisional (also called informational)
200–299	Success
300–399	Redirection
400–499	Client error
500–599	Server error
600–699	Global failure

Apart from the start line (status line in responses and request line in requests) the format of requests and responses is identical, as shown in Figure 3.8. Section 3.4 describes the format of the request line. Sections 3.6 and 3.7 describe the format of the rest of the message (header fields and message body, respectively).

3.6 Header Fields

Right after the start line, SIP messages (both requests and responses) contain a set of header fields (see Figure 3.8). There are mandatory header fields that appear in every message and optional header fields that only appear when needed. A header field consists of the header field's name, a colon, and the header field's value, as shown in the example below:

```
To: Alice Smith <sip:Alice.Smith@domain.com>;tag=1234
```

The value of a header field can consist of multiple items. The `To` header field above contains a display name (Alice Smith), a URI

```
sip:Alice.Smith@domain.com
```

and a `tag` parameter.

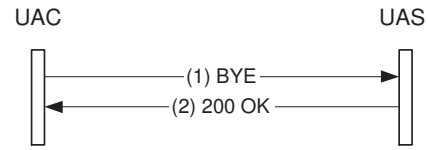

Figure 3.9. SIP transaction

Some header fields can have more than one entry in the same message, as shown in the example below:

```
Route: <sip:p1.domain1.com>
Route: <sip:p34.domain2.com>
```

Multi-entry header fields can appear in a single-value-per-line form, as shown above, or in a comma-separated value form, as shown below. Both formats are equivalent.

```
Route: <sip:p1.domain1.com>, <sip:p34.domain2.com>
```

There are six mandatory header fields that appear in every SIP message. They are `To`, `From`, `Cseq`, `Call-ID`, `Max-Forwards`, and `Via`.

The `To` header field contains the URI of the destination of the request. However, this URI is not used to route the request. It is intended for human consumption and for filtering purposes. For example, a user can have a private URI and a professional URI and requests can be filtered depending on which URI appears in the `To` field. The `tag` parameter is used to distinguish, in the presence of forking proxies, different user agents that are identified with the same URI.

The `From` header field contains the URI of the originator of the request. Like the `To` header field, it is mainly used for human consumption and for filtering purposes.

The `Cseq` header field contains a sequence number and a method name. They are used to match requests and responses.

The `Call-ID` provides a unique identifier for a SIP message exchange.

The `Max-Forwards` header field is used to avoid routing loops. Every proxy that handles a request decrements its value by one, and if it reaches zero, the request is discarded.

The `Via` header field keeps track of all the proxies a request has traversed. The response uses these `Via` entries so that it traverses the same proxies as the request did in the opposite direction.

3.7 Message Body

As Figure 3.8 shows, the message body is separated from the header fields by an empty line. SIP messages can carry any type of body and even multipart bodies using MIME (Multipurpose Internet Mail Extensions) [42] encoding.

SIP uses MIME to encode its message bodies [17]. Consequently, SIP bodies are described in the same way as attachments to an email message. A set of header fields provide information about the body: its length, its format, and how it should be handled. For example, the header fields below describe the SDP session description of Figure 3.1:

```
Content-Disposition: session
Content-Type: application/sdp
Content-Length: 193
```

The `Content-Disposition` indicates that the body is a session description, the `Content-Type` indicates that the session description uses the SDP format, and the `Content-Length` contains the length of the body in bytes.

Figure 3.10 shows an example of a multipart body encoded using MIME. The first body part is an SDP session description and the second body part consists of the text "This is the second body part". Note that the `Content-Type` for the whole body is `multipart/mixed` and that each body part has its own `Content-Type`, namely `application/sdp` and `text/plain`.

An important property of bodies is that they are transmitted end-to-end. That is, proxies do not need to parse the message body in order to route the message. In fact, the user agents may choose to encrypt the contents of the message body end-to-end. In this case, proxies would not even be able to tell which type of session was being established between both user agents.

3.8 Message Encoding

SIP uses text encoding as opposed to binary for its messages. When SIP was being designed, which encoding was best was discussed at length. Text proponents claimed that text-based protocols are debugged more easily because they can be read directly by a human and that text protocols are more flexible and easier to extend with new features. Binary proponents argued that binary protocols use bandwidth more efficiently and are

```
Content-Type: multipart/mixed; boundary="0806040504000805090"
Content-Length: 384

--0806040504000805090
Content-Type: application/sdp
Content-Disposition: session

v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.2.1
s=Let's talk about swimming techniques
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=sendrecv
m=video 20002 RTP/AVP 31
a=sendrecv
--0806040504000805090--
Content-Type: text/plain

This is the second body part
--0806040504000805090--
```

Figure 3.10. MIME encoding of a multipart body

easy to debug and extend with the proper tools. Both types of encoding have advantages and disadvantages.

A textual encoding was chosen for SIP mostly because there are more designers and developers who are familiar with how to use and extend text-based protocols. Therefore, it was believed that a textual encoding would result in more services being created. For example, the tools needed to create HTTP-based services are taught at many university courses. The same or similar tools (e.g., CGI (Common Gateway Interface) for SIP [78] and SIP Servlets [103]) can be used to create SIP-based services as well.

SIP's textual encoding can result in relatively large messages, which may be problematic on low-bandwidth interfaces. SIP messages can be compressed [12] using the SigComp [102] signalling compression mechanism.

3.9 SIP Transactions

SIP defines three types of transactions: regular transactions, INVITE–ACK transactions, and CANCEL transactions. The type of a particular transaction depends on the request initiating it.

Regular transactions are initiated by any request except INVITE, ACK, or CANCEL. Figure 3.11 shows a regular BYE transaction. In a regular transaction, the user agent server receives a request and generates a final response that terminates the transaction.

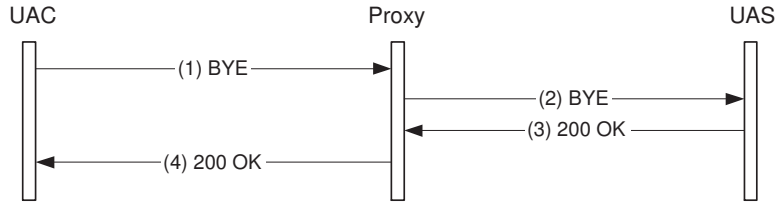

Figure 3.11. Regular transaction

An INVITE–ACK transaction involves two transactions: an INVITE transaction and an ACK transaction, as shown in Figure 3.12. The user agent server receives an INVITE request and generates zero or more provisional responses and a final response. When the user agent client receives the final response, it generates an ACK request, which does not have any response associated with it.

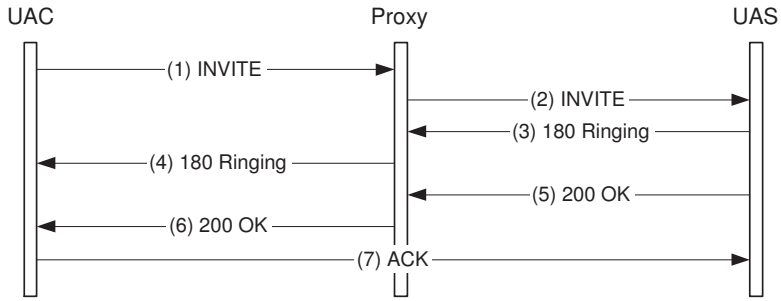

Figure 3.12. INVITE–ACK transaction

CANCEL transactions are initiated by a CANCEL request and are always connected to a previous transaction (i.e., the transaction to be cancelled). CANCEL transactions are similar to regular transactions, with the difference that the final response is generated by the next SIP hop (typically a proxy) instead of by the user agent server. Figure 3.13 shows a CANCEL transaction cancelling an INVITE transaction. Note that the INVITE transaction, once it is cancelled, terminates as usual (i.e., final response plus ACK).

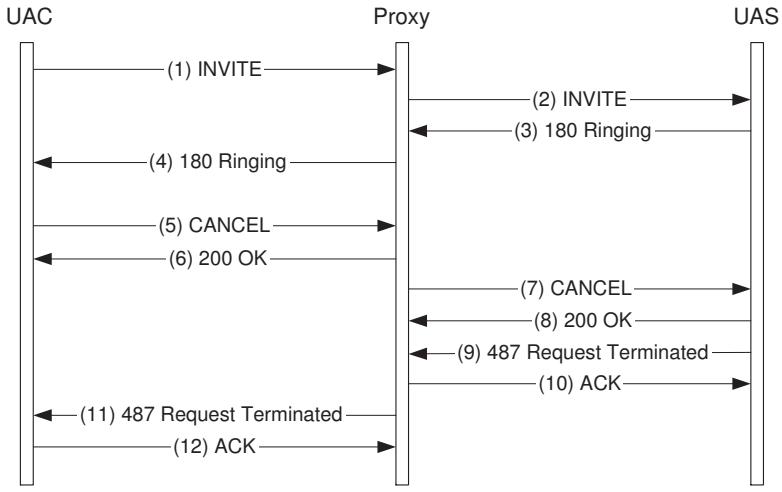

Figure 3.13. CANCEL transaction

Figure 3.14. Alice registers her location

3.10 Message Flow for Session Establishment

This is an example of how to use SIP to establish a multimedia session. First of all, Alice registers her current location

```
sip:alice@pda.com
```

with the registrar at domain.com, as shown in Figure 3.14. To do this, Alice sends a REGISTER request (Figure 3.15) indicating that requests addressed to the URI in the `To` header field

```
sip:Alice.Smith@domain.com
```

should be relayed to the URI in the `Contact` header field

```
sip:alice@pda.com
```

The *Request-URI* of the REGISTER request contains the domain of the registrar (domain.com). The registrar responds with a 200 (OK) response (Figure 3.16) indicating that the transaction was successfully completed.

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.1:5060;branch=z9hG4bKna43f
Max-Forwards: 70
To: <sip:Alice.Smith@domain.com>
From: <sip:Alice@pda.com>;tag=453448
Call-ID: 843528637684230998sdasdsfgt
Cseq: 1 REGISTER
Contact: <sip:alice@pda.com>
Expires: 7200
Content-Length: 0
```

Figure 3.15. (1) REGISTER

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.1:5060;branch=z9hG4bKna43f
   ;received=192.0.2.1
To: <sip:Alice.Smith@domain.com>;tag=54262
From: <sip:Alice@pda.com>;tag=453448
Call-ID: 843528637684230998sdasdsfgt
Cseq: 1 REGISTER
Contact: <sip:alice@pda.com>;expires=7200
Date: Sat, 25 Mar 2006 17:38:00 GMT
Content-Length: 0
```

Figure 3.16. (2) 200 OK

At a later time, Bob invites Alice to an audio session. Figure 3.17 shows the establishment of the audio session between Bob and Alice through the proxy server at domain.com.

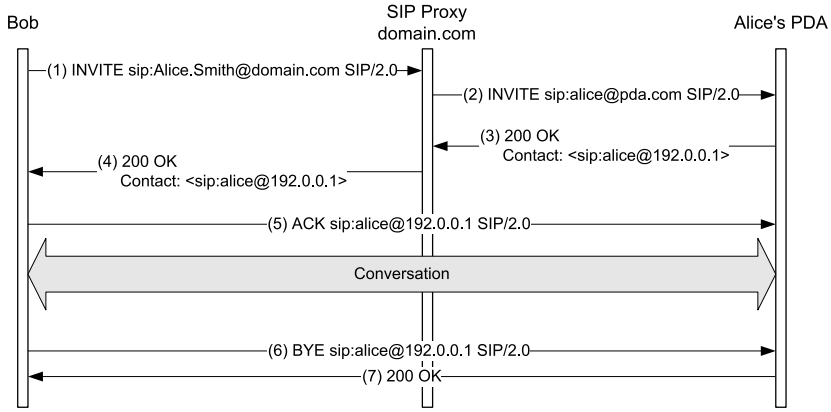

Figure 3.17. Session establishment through a proxy

Bob sends an INVITE request (Figure 3.18) using Alice's public URI

`sip:Alice.Smith@domain.com`

as the *Request-URI*. The proxy at domain.com relays the INVITE request to Alice at her current location (her PDA). Alice accepts the invitation sending a 200 (OK) response, which is relayed by the proxy to Bob (Figure 3.19).

Note that Alice has included a `Contact` header field in her 200 (OK) response. This header field is used by Bob to send subsequent messages to Alice. This way, once the proxy at domain.com has helped Bob locate Alice, Bob and Alice can exchange messages directly between them.

```
INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576sl
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Type: application/sdp
Content-Length: 138

v=0
o=bob 2890844526 2890844526 IN IP4 ws1.domain2.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 3.18. (1) INVITE

Bob uses the URI in the `Contact` header field of the 200 (OK) response to send his ACK. Now that the session (i.e., an audio stream) is established, Bob and Alice can talk with each other. If, in the middle of the session, they wanted to make any changes to the session (e.g., add video), all they would need to do would be to issue another INVITE request with an updated session description. INVITE requests sent within an ongoing session are usually referred to as re-INVITEs [19]. (UPDATE requests can also be used to modify ongoing sessions. In any case, UPDATEs are used when no interactions with the callee are expected. In this case, we use re-INVITE because the callee is typically prompted before adding video to a session.)

When Bob and Alice finish their conversation, Bob sends a BYE request to Alice. Note that, as with the ACK, this request is sent directly to Alice, without the intervention of the proxy. Alice responds with a 200 (OK) response to the BYE request.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
   ;received=192.0.100.2
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576sl
To: Alice <sip:Alice.Smith@domain.com>;tag=1df345fkj
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:alice@192.0.2.1>
Content-Type: application/sdp
Content-Length: 132

v=0
o=alice 2890844545 2890844545 IN IP4 192.0.2.1
s=-
c=IN IP4 192.0.2.1
t=0 0
m=audio 30000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 3.19. (4) 200 OK

3.11 SIP Dialogs

In Figure 3.17, Bob and Alice exchange a number of SIP messages in order to establish (and terminate) a session. The exchange of a set of SIP messages between two user agents is referred to as a SIP dialog. In our example the SIP dialog is established by the INVITE transaction and is terminated by the BYE transaction. Note, however, that, in addition to INVITE, there are other methods that can create dialogs as well (e.g., SUBSCRIBE).

When a SIP dialog is established (e.g., with an INVITE transaction), all the subsequent requests within that dialog follow the same path. In our example, all the requests after the INVITE (the ACK (5) and the BYE (6)) are sent end-to-end between the user agents. However, some proxies choose to remain in the signaling path for subsequent requests within a dialog instead of routing the first INVITE request and stepping down after the 200 (OK) response. Let us study the mechanism used by proxies to stay in the path after the first INVITE request. It consists of three header fields: `Record-Route`, `Route`, and `Contact`.

Figure 3.20 shows a message flow where the proxy at domain.com remains in the path for all the requests sent within the dialog. The proxy requests to remain in the path by adding a `Record-Route` header field to the INVITE request (2). The `lr` parameter that appears at the end of the URI indicates that this proxy is RFC 3261-compliant (older proxies used a different routing mechanism).

Alice obtains the `Record-Route` header field with the proxy's URI in the INVITE request (2), and Bob obtains it in the 200 (OK) response (4). From that point on, both Bob and Alice insert a `Route` header field in their requests, indicating that the proxy at domain.com needs to be visited. The ACK (5 and 6) is an example of a request with a `Route` header field sent from Bob to Alice. The BYE (7 and 8) shows that requests in the opposite direction (i.e., from Alice to Bob) use the same `Route` mechanism.

3.12 Event Notification

So far, we have seen how to use SIP to establish sessions. Additionally, SIP can also be used to obtain the status of a given resource and track changes in that status. For example, at a given moment, the state of

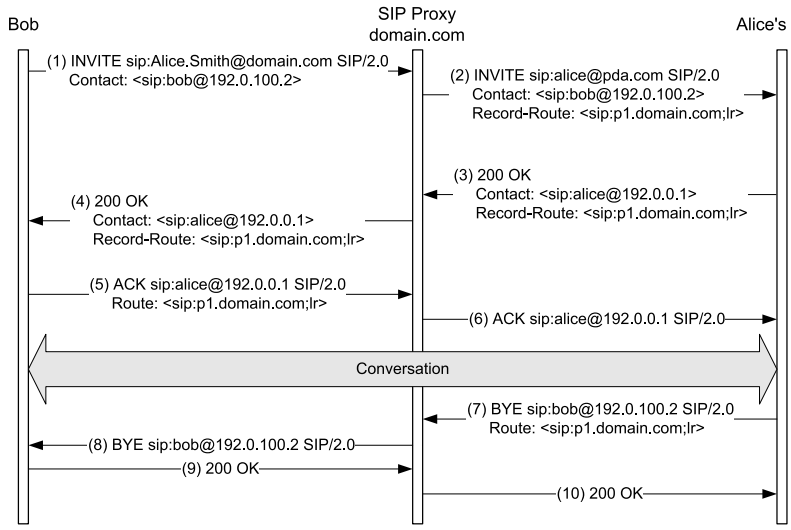

Figure 3.20. Usage of `Record-Route`, `Route`, and `Contact`

Alice's presence is "online". When she logs off from her computer her presence status changes to "offline". In this example the resource is Alice and the status information is her presence information.

SIP includes a framework for event notification [111]. It uses the SUBSCRIBE and NOTIFY methods. The entity interested in the status information of a resource *subscribes* to that information. The entity that keeps track of the resource state will send a NOTIFY request with the current status information of the resource and a new NOTIFY request every time the status changes. The type of status information is defined by an `Event` header field. Specifications defining new values for the `Event` header field are called *event packages*.

The event notification framework defines two new roles in SIP: the *subscriber* and the *notifier*. A subscriber is a SIP UA that sends a SUBSCRIBE request for a particular event. A subscriber gets NOTIFY requests containing state information related to the subscribed event. A notifier is a SIP UA that receives SUBSCRIBE requests for a particular event and generates a NOTIFY request containing the state information related to the subscribed event. A notifier keeps a subscription state for each of the subscribers.

Figure 3.21 shows an example where Alice, acting in the role of a subscriber, subscribes to her voicemail [84]. The voicemail server is acting as a notifier. In this case, the status information she is interested in is the number of messages that have arrived at the voicemail. This corresponds

to an Event header field of value message-summary.

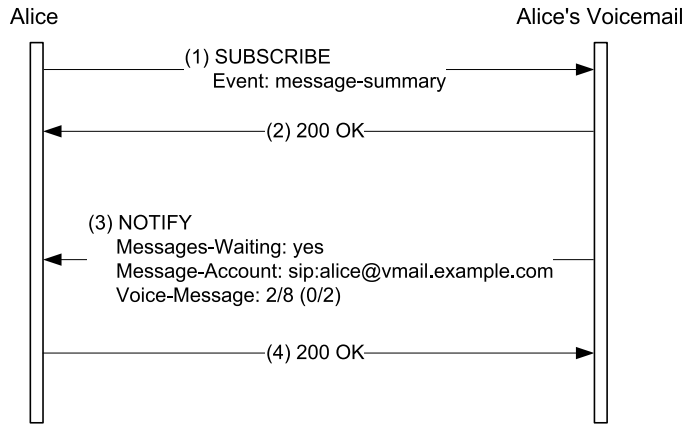

Figure 3.21. Voicemail status information

Note that the 200 (OK) response (2) to the SUBSCRIBE request only indicates that the SUBSCRIBE transaction has been successful. That is, the subscription has been accepted by the voicemail. The information about the resource always arrives in a NOTIFY transaction. The NOTIFY request (3) in Figure 3.21 shows the body of the NOTIFY. It indicates that Alice's voicemail has two new messages and eight old messages, of which none of the new and two of the old messages are urgent (figures enclosed in parentheses).

If, before the subscription expires, Alice's voicemail receives a new message, it will send a new NOTIFY request to Alice informing her about the new arrival.

3.13 Extending SIP

SIP's extension negotiation mechanism uses three header fields: Supported, Require, and Unsupported. When a SIP dialog is being established the user agent client lists all the names of the extensions it wants to use for that dialog in a Require header field, and all the names of the extensions it supports not listed previously in a Supported header field. The names of the extensions are referred to as *option tags*.

The user agent server inspects the Require header field and, if it does not support any of the extensions listed there, it sends back an error response indicating that the dialog could not be established. This error re-

sponse contains an `Unsupported` header field listing the extensions the user agent server did not support.

If the user agent server supports (and is willing to use) all the required extensions, it should decide whether or not it wants to use any extra extension for this dialog and, if so, it includes the option tag for the extension in the `Require` header field of its response. If this option tag was included in the `Supported` header field of the client, the dialog will be established. Otherwise, the client does not support the extension (or is not willing to use it). In this case the user agent server includes the extension which is required by the server in a `Require` header field of an error response. Such an error response terminates the establishment of the dialog.

Figure 3.22 shows a successful extension negotiation between Bob and Alice. They end up using the extensions whose option tags are `foo1`, `foo2`, and `foo4`.

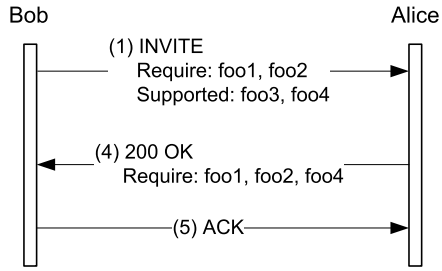

Figure 3.22. Extension negotiation in SIP

In addition to option tags, SIP can be extended by defining new methods. We saw in Table 3.2 that there are many SIP methods, but that the core protocol only uses a subset of them. The rest of the methods are defined in SIP extensions.

In a SIP dialog the user agents need to know which methods the other end understands. For this purpose, each of the user agents includes an `Allow` header field in its messages listing all the methods it supports. An example of an `Allow` header field is

```
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
```

As we can see, the `Allow` header field lets user agents advertise the methods they support, but it cannot be used to express the fact that a particular method is required for a particular dialog. To provide such functionality, an option tag associated with the method required is defined. This way a user agent can include the option tag in its `Require`

header field and force the remote end to apply the extension and, so, to understand the method. The extension for reliable provisional responses [122] is an example of an option tag associated with a method: PRACK.

3.14 NAT Traversal

Originally, the offer/answer model [121] only supported the establishment of sessions between user agents in the same address space. The deployment of NATs on the Internet and on many private networks made it necessary to include a NAT traversal mechanism in SIP. ICE (Interactive Connectivity Establishment) [116] is a probe-based NAT traversal methodology that uses NAT traversal protocols such as STUN [120] (Simple Traversal of User Datagram Protocol) and TURN [85] (Traversal Using Relays around NAT). User agents establishing a session use these protocols to obtain a list of IP addresses where they could potentially be reachable. The user agents exchange their address lists and use STUN-based connectivity checks to discover which address pair or pairs can be actually used for the session. Once the session is established, the user agents send connectivity checks periodically so that the NAT bindings being used for the session do not expire.

4. Framework Overview

This thesis describes a framework to provide SIP-based multimedia communication services. Applications and services built on top of this framework take advantage of the functionality provided by the framework, whose structure is shown in Figure 4.1.

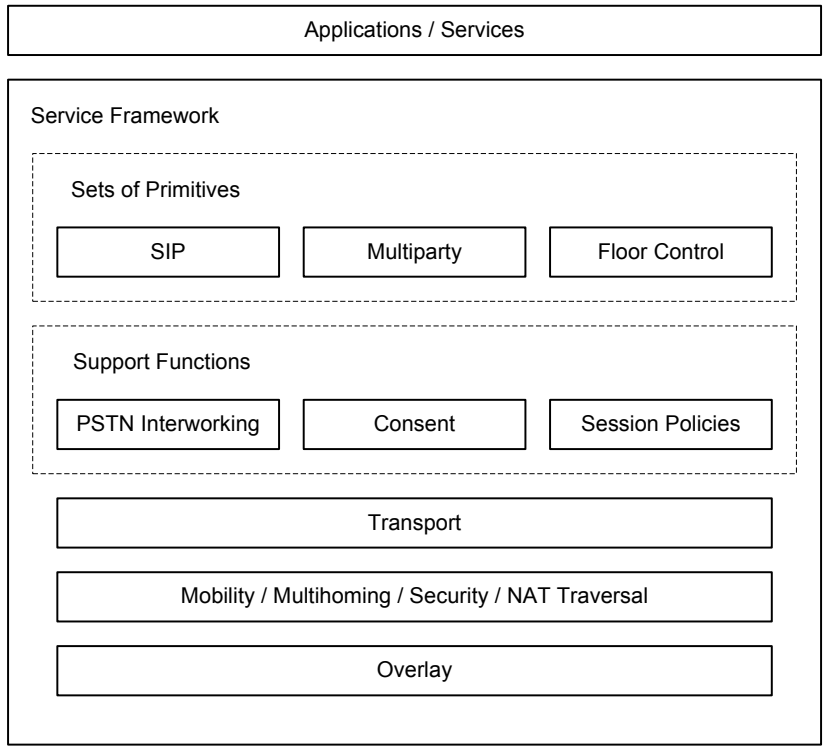

Figure 4.1. Framework structure

The framework includes functionality that is likely to be used by many applications and services. In this way, different individual applications and services can rely on the functionality provided by the framework instead of implementing it themselves. Reusing the same functionality

across different applications and services results in reducing their implementation cost. Figure 4.1 shows the structure of the framework. As stated in Section 2.1, an intuitive way to think about a framework and its elements is as a toolkit and the tools inside it. The framework provides applications and services with tools. Different applications and services will use different tools within the toolkit. A given service may not use a particular tool, which may be used by a different service.

Some of the tools provided by the framework can be grouped, as shown in Figure 4.1. Some tools consist of sets of primitives and some other tools consist of support functions. Figure 4.1 helps visualizing the framework and its components. In general, tools upper in the figure make use of tools lower in the figure. However, the position of particular tools in Figure 4.1 could be debated. For example, SIP appears higher than Session Policies in the figure because Session Policies will be used as part of the session establishment process performed by SIP. A different (but also valid) perspective would be to place Session Policies higher in the figure given that Session Policies use SIP as their transport. Therefore, the position of the tool in the figure is not overly important given that some tools within the framework make use of other tools within the framework as part of their operations. This framework is not an architecture with a rigid structure. As discussed earlier, this framework is a set of tools that can be combined in different ways to provide applications and services with functionality needed to implement a wide variety of multimedia communication services.

In addition to the tools provided by the framework, services and applications can make use of additional tools, which may not be included in the framework. As stated before, this framework is not a rigid architecture that could limit what applications and services could use. Application and services are free to use any functionality they need outside this framework. As discussed in Chapter 1, we have only included in the framework functionality that is related to either our scientific work or to our engineering work.

The tools in the framework are described at the protocol model level [110]. In addition, the RFCs associated with each particular tool discuss its operation in full detail. An application designer needs to combine several of these tools into a system to build an application or service. Our framework provides application designers with the information they need to build their systems.

Note that implementers of such systems can work at several different levels. The implementer of a software library such as a SIP stack will use the SIP specification [124] as the basis for the stack. However, the implementer of a VoIP client may simply use an existing SIP stack that hides all SIP-related details behind an API. We do not discuss this type of API in this thesis.

System architects can also take tools from our framework and include them in their system architectures. For example, the IMS (IP Multimedia Subsystem) [18] uses some of the primitives (e.g., SIP and Multiparty) and support functions (e.g., PSTN Interworking) in our framework.

4.1 Sets of Primitives

As shown in Figure 4.1, the framework includes sets of primitives related to SIP, to multiparty communications, and to floor control (i.e., the ability to manage the access to a shared resource). Given that the framework described in this thesis is based on SIP, it includes the primitives SIP provides. These primitives include, for example, establishing a session with a remote user agent.

The sets of primitives intended to establish multiparty communications involve the use of URI-list services. URI-list services are specially useful to establish ad-hoc conference calls in an efficient manner. A user agent provides the URI-list server with a set of recipients and the URI-list server establishes a conference call among all the recipients and the original user agent. The use of a URI-list service reduces the amount of signalling on the user agent's access network. That is why URI-list services are especially suitable to wireless devices with bandwidth or delay restrictions over the air interface.

In multiparty communications, it is often important to provide some type of floor control in order to regulate which user agent can access a given resource (e.g., send media) at a given time. BFCP (Binary Floor Control Protocol) is a floor control protocol, which uses binary encoding. Its compact binary encoding makes BFCP also especially suitable to wireless devices with bandwidth or delay restrictions over the air interface.

The sets of primitives included in the framework are discussed in Chapter 5.

4.2 Support Functions

As shown in Figure 4.1, the framework includes support functions related to PSTN interworking, consent-based communications, and session policies. Being able to communicate with devices on the PSTN is a key requirement for most SIP-based communication networks. PSTN-SIP gateways perform protocol translations at the edge of the networks to make such interworking possible.

Consent-based communications are an important tool to avoid unwanted communications (e.g., SPIT) and certain types of DoS attacks. A SIP network implementing consent-based communications does not forward communication requests to their recipients unless the recipients have explicitly given his or her consent to receive such communication requests.

Session policies allow user agents to maintain the end-to-end principle [127] while taking into consideration restrictions network operators may impose on their sessions. Network operators explicitly inform user agents about such restrictions so that the user agents themselves can modify their communication sessions to suit their communication needs as well as possible while observing the restrictions received.

The support functions included in the framework are discussed in Chapter 6.

4.3 Transport

As shown in Figure 4.1, the framework includes a transport function. The transport of SIP messages between user agents and proxy servers and between proxy servers has different requirements. In particular, large proxy servers exchanging SIP traffic between them face the HOLB (Head of the Line Blocking) problem, which is not typically a problem for user agents exchanging SIP traffic with their proxy server. The use of SCTP [134] between proxy servers eliminates HOLB and, thus, decreases the delays seen by user agents establishing sessions.

The transport function included in the framework is discussed in Chapter 7.

4.4 Mobility, Multihoming, Security, and NAT Traversal

As shown in Figure 4.1, the framework includes functions related to mobility, multihoming, security, and NAT traversal. In the framework, these functions are provided by HIP (Host Identity Protocol). HIP provides all these functions in an integrated way so that all the functions can be provided at the same time in a coherent way. The provision of functions such as NAT traversal at lower layers (as opposed to providing them at the application layer) makes it possible to handle different application flows in an integrated fashion. This integrated flow handling can decrease the delays seen by user agents when establishing sessions in some scenarios.

The mobility, multihoming, security, and NAT traversal functions included in the framework are discussed in Chapter 8.

4.5 Overlays

As shown in Figure 4.1, the framework includes functions related to overlays. An overlay is a network that is built on top of another network (e.g., a peer-to-peer overlay that runs on top of the Internet). The framework also uses HIP to build overlays in situations where an infrastructure of SIP servers is not available. The ID/locator split implemented by HIP makes it a natural choice to build secure overlays. Additionally, the integrated flow handling mentioned in Section 4.4 is especially relevant for reducing session establishment delays in overlay scenarios.

The functions related to overlays included in the framework are discussed in Chapter 9.

5. Main Primitives

Figure 4.1 in Chapter 4 shows the structure of the framework described in this thesis. The framework includes sets of primitives related to SIP, to multiparty communications, and to floor control. All these primitives are discussed in this section.

As discussed in Section 2.3, when we use the term primitives we refer to protocol primitives. In practice, some applications do not access the protocol primitives directly. Instead, they access the protocol primitives through different APIs (e.g., JSR 32 [105], JSR 141 [104], JSR 180 [108], JSR 281 [107], JSR 289 [106], or Parlay X [3]). We do not discuss these APIs in this thesis.

This section is based on our work on the main SIP specification, which is documented in [124], our work on URI-list services, which is documented in [29], [43], [44], [21], [30], and [24], and our work on BFCP (Binary Floor Control Protocol), which is documented in [26], [13], and [14].

In [29], we describe a framework for URI-list services and discuss the security implications of using them in the network. In [43], we specify 'To', 'Cc', and 'Bcc' functionality (which is typical in email) for SIP. In [44], [21], [30], and [24], we specify URI-list services for MESSAGE, INVITE, SUBSCRIBE, and REFER requests, respectively.

In [26], we specify BFCP. In [13], we specify how to establish a BFCP connection using an SDP offer/answer exchange. In [14], we specify how to establish a BFCP connection outside an SDP offer/answer exchange.

5.1 SIP Primitives

SIP, as specified in [124], provides primitives to establish, modify, and terminate multimedia communication sessions. Additionally, SIP can also

be used to add and remove participants to and from existing sessions. SIP is mostly used to create and manage unicast point-to-point media sessions. When SIP is used to invite participants to multicast sessions, protocols different than SIP are used to create such sessions or to manage them. Chapter 3 provides further detail on SIP's basic functionality.

5.2 Multiparty Communications

SIP natively supports multiparty communications. For example, a user agent can invite a number of participants to a particular session by sending an INVITE request to each of them. The user agent can also send an instant message to all those participants by sending a MESSAGE request to each of them.

Performing several transactions like the ones just described works well when the number of recipients is low and the size of the request is small. However, when the number of recipients is high or the request is large the access network of the user agent needs to carry a considerable amount of traffic. Completing all the transactions on a low-bandwidth access can take a long time. This issue is particularly relevant to terminals using low-bandwidth radio accesses such as GPRS (General Packet Radio Service) [4].

In the email world, this issue is directly addressed in SMTP (Simple Mail Transfer Protocol) [74]. Clients can send a single email message with multiple recipients so that the MTAs (Mail Transfer Agents) in the network deliver it to the different receivers. SMTP also supports mailing lists through address expansions in the network.

Our proposal to resolve this issue in SIP is to introduce URI-list services [29] in the network. The task of a SIP URI-list service is to receive a request that contains or references a URI list (i.e., a list of one or more URIs) and send a number of similar requests to the destinations in this list. A given URI-list service can take as an input a URI list contained in the SIP request sent by the client or an external URI list (e.g., the Request-URI is a SIP URI that is associated with a URI list at the server). External URI lists are usually encoded using the XML (Extensible Markup Language) format for representing resource lists [114] and are typically set up using out-of-band mechanisms (e.g., XML Configuration Access Protocol (XCAP) [115]). A server providing URI-list ser-

vices for subscriptions is usually referred to as an RLS (Resource List Server) [112].

Figures 5.1 and 5.2 show a user sending the same instant message to three different recipients. In Figures 5.1, the sender performs three different MESSAGE transactions. In Figures 5.2, the sender performs a single MESSAGE transaction with a server providing URI-list services. The server, in turn, performs a different MESSAGE transaction with each of the recipients. The MESSAGE request from the original sender to the server contains, in addition to the instant message, a list with the recipients of the instant message. Each recipient can be a 'To', 'Cc', or 'Bcc' recipient [43].

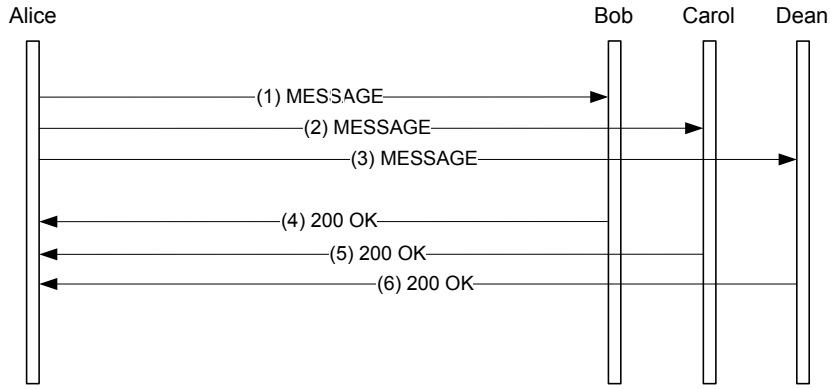

Figure 5.1. Multiple MESSAGE requests

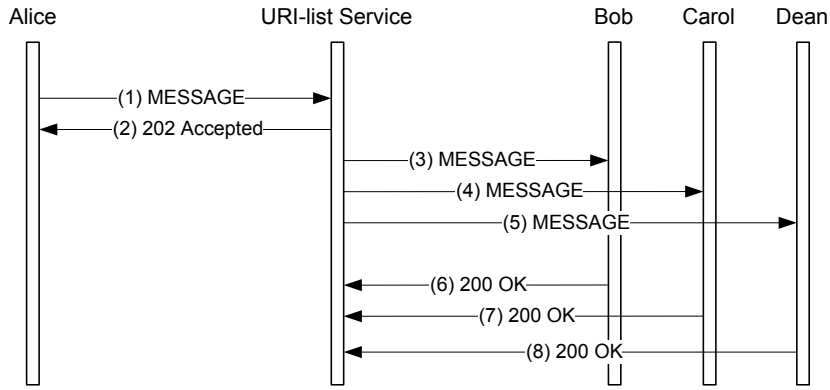

Figure 5.2. Multiple-recipient MESSAGE request

If a user agent using a URI-list service is interested in the status of the transactions performed by the server towards the final recipients, the user agent uses different mechanisms depending on the type of request being

handled. If the request is a MESSAGE request, the user agent may use a delivery notification mechanism. If the request is an INVITE request, the user agent can use the conference event package [125].

URI-lists are used to provide several services. For example, in order to meet its requirements [97], the OMA (Open Mobile Alliance) PoC (Push to talk over Cellular) service (which uses a SIP-based architecture [95]) uses URI-list services on its control plane [96].

5.3 Floor Control

Floor control provides the ability to manage the access to a shared resource. Examples of shared resources in a multiparty communications session are audio streams, video streams, and shared virtual whiteboards. Floor control is a key requirement for some types of sessions such as large conference calls where many participants are likely to attempt to talk at the same time. Floor control is also required in one-to-one half-duplex communication sessions where only one of the participants can send media at a given time.

While a floor control mechanism indicates which participant can access a given resource at a give time, enforcing that only that participant can actually access the resource is not considered part of the floor control mechanism. There are many situations where such an enforcement is not needed. For example, in a face-to-face meeting among a few people, the chair of the meeting gives the floor to a given participant so that the participant can talk but the chair does not prevent in any way other participants from talking at the same time. It is assumed that the participants play by the rules and will follow the instructions given by the chair. There are also situations where such enforcement is needed. Depending on the nature of the communication session, there are different mechanisms to enforce floor control related policies. For example, a conference bridge can be configured to ignore inputs from participants who do not hold the floor at the time.

The area of floor control has been researched in the past. Depaoli et al. [36] describe a multimedia conferencing system that includes floor control. Dommel el. at. [38] describe a set of requirements on floor control in networked multimedia applications. Handley et al. [50] describe an architecture for multimedia conferencing on the Internet that also in-

cludes floor control. Koskelainen at. al. [77] describe a conference control framework based on SIP where floor control is implemented using SOAP (originally defined as Simple Object Access Protocol, although the term is not considered an acronym any longer) commands. A multipoint data communication service for use in multimedia conferencing environments is described in [65]. This service includes a token-based system to implement exclusive access to resources [62]. Token allocations can be performed within a conference [66] to implement floor control functionality. This communication service was designed to be able to run on different networks. However, guaranteeing an acceptable performance when running it over a best-effort IP service was considered to be difficult [49]. The floor control part of the service also had scalability problems [98]. Integrating this floor control mechanism with SIP was considered to be too heavy since the system also included functionality, such as capability negotiation, which was already present in SIP.

Our proposal to implement floor control is BFCP [9] (Binary Floor Control Protocol), which meets the general requirements on floor control protocols [76]. When designing BFCP, we chose a binary encoding (as opposed to other encoding such as XML) so that BFCP could be used in low-bandwidth environments with tough timing requirements. We also designed BFCP in a modular way so that it could be easily integrated in SIP-based conferencing systems. We chose a light-weight design where only fundamental floor control primitives are supported and advanced functionality is implemented at higher levels. These design choices separate BFCP from past floor control protocol proposals.

Figure 5.3 shows the logical architecture BFCP follows. Floor participants request a floor from the floor control server. The floor control server consults the chair of that floor so that the floor chair makes a decision. When the decision is made (e.g., granting the floor), the floor control server notifies the relevant participants.

The floor chair logical function can also be implemented as part of the floor control server. The floor control server can follow a pre-defined policy to handle floor requests in an automatic fashion. For example, the floor control server can implement a FIFO (First In First Out) queue for incoming floor requests and grant the floor to the request leaving the queue at any point.

Figure 5.4 shows a typical BFCP message flow. A participant sends a FloorRequest message (1) to request a given floor. The floor control

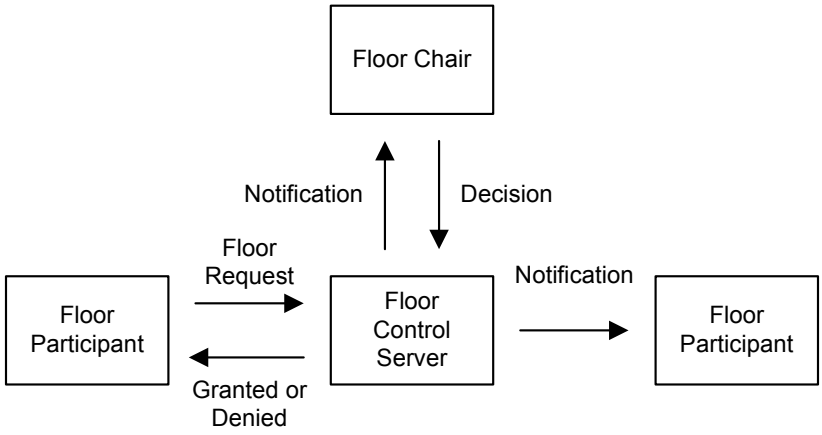

Figure 5.3. BFCP logical architecture

server informs the participant about the status of the floor request using FloorRequestStatus messages. Eventually, the participant gets the floor (4). When the participant has finished using the shared resource (e.g., sending media), the participant sends a FloorRelease message (5).

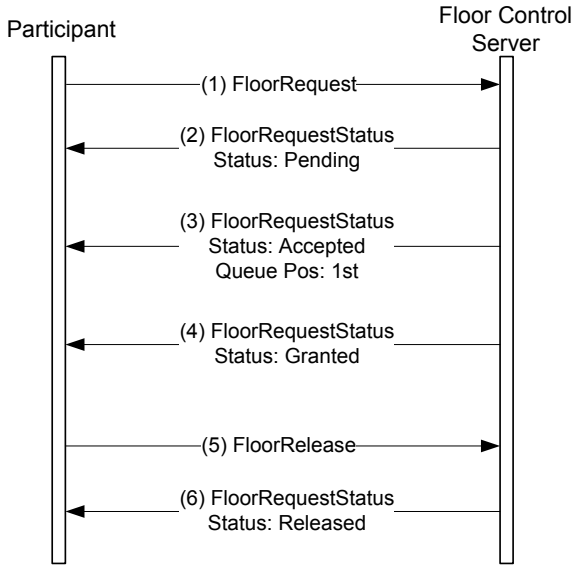

Figure 5.4. BFCP message flow

BFCP is used in many video conferencing systems at present. Some of these systems place the floor control server functionality on a node behind a NAT. In order to improve its NAT traversal properties, some systems transport BFCP over UDP or HTTP instead of over TCP, which is its standard transport. In the future, there may be standard extensions to

BFCP that allow transporting it over different transport protocols, as we had originally envisioned when originally designing BFCP.

6. Support Functions

Figure 4.1 in Chapter 4 shows the structure of the framework described in this thesis. The framework includes support functions related to PSTN interworking, to consent-based communications, and to session policies. All these support functions are discussed in this chapter.

This chapter is based on our work on PSTN interworking, which is documented in [27], and [28], our work on session policies, which is documented in Publication I, [57], [59], [58], and [61], and our work on consent-based communications, which is documented in Publication II, [117], [118], [15], and [16].

In [27], we specify a mapping between ISUP (ISDN User Part Signaling) [63] and SIP. In [28], we specify how to map ISUP overlap signalling to SIP.

In Publication I, we propose a new policy control mechanism for SIP, which we refer to as session policies. We discuss the advantages of this approach over other existing policy control mechanisms such as the one used in the IMS (IP Multimedia Subsystem) [18], which is based on inspecting and modifying session descriptions in the network. We also describe how to integrate session policies in the IMS architecture. Finally, we describe our experience implementing a proof-of-concept prototype of the proposed mechanism. In [57], we describe some of the functions typically performed by SBCs (Session Border Controllers). Some of these functions can be implemented in a more end-to-end fashion by using session policies instead. In [59], we describe a framework for session policies in SIP. In [58], we specify an event package to implement session-specific policies. In [61], we specify an XML-based language to describe media-related session policies

In Publication II, we propose a framework to add consent-based communications to SIP. We discuss the properties of the framework and its

applicability. We also evaluate the framework. The evaluation is based on our implementation of the framework in a proof-of-concept prototype. In [117], we discuss the requirements to implement consent-based communications in SIP. In [118], we describe a framework to implement consent-based communications in SIP. In [15], we specify an XML-based document format to request consent. In [16], we specify a SIP event package to be used by SIP relays to inform user agents about the consent-related status of the entries to be added to a resource list.

6.1 PSTN Interworking

As discussed in Section 2.4.1, interworking with the PSTN is currently essential for any network providing communication services. In the early stages of SIP's development, Schulzrinne et al. identified the need for SIP systems to interwork with the PSTN through gateways [130].

There are many signalling protocols used on the PSTN. Some of them are used on the UNI (user-to-network interface) while others are used on the NNI (network-to-network interface). When introducing a new protocol such as SIP, interworking over the NNI is more important because through the NNI interface it is possible to reach any PSTN user. The interworking over the UNI interface is important to connect islands using PSTN protocols (e.g., a company's PBX (Private Branch Exchange) with another company's PBX) between them. This way of connecting PSTN islands using SIP is generally referred to as SIP trunking. We have focused on the interworking between ISUP (ISDN User Part Signaling) [63] and SIP because ISUP is the most widely used protocol over the NNI (network-to-network interface).

We proposed an architecture for ISUP to SIP interworking [10]. Additionally, we specified how to map ISUP signalling to SIP [27] [28]. The two most common interworking architectures at present, SIP-T [140] and SIP-I [64], were based on our work. Figure 6.1 shows the logical architecture of a PSTN to SIP gateway.

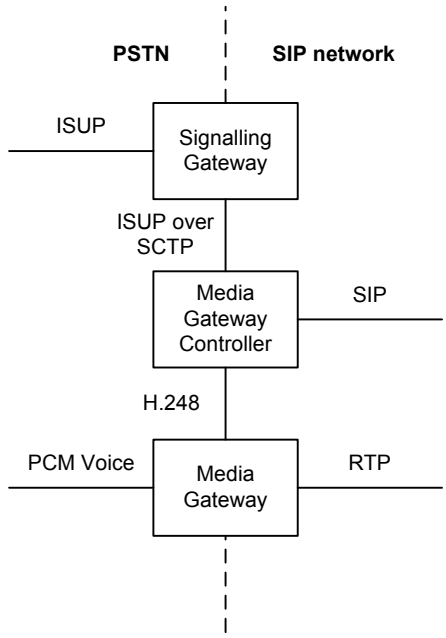

Figure 6.1. PSTN-to-SIP gateway architecture

6.2 Session Policies

As discussed in Chapter 3, SIP uses the offer/answer model [121] to establish sessions. The offer/answer model was originally designed to allow user agents to agree on the parameters to be used for a given session. That is, it was assumed that the user agents were the only entities involved in that negotiation. Proxy servers were designed to simply help route messages between the user agents. Proxy servers did not play any role in the offer/answer model and, thus, did not need to access the session descriptions being exchanged between the user agents.

As SIP got deployed, the assumption above did not hold in many environments. The operators of the networks between the user agents wanted to be part of session negotiations. For example, the operator of a cellular network might want to carry different media streams (e.g., audio and video) in different radio bearers or might prefer the use of a particular codec over a different one. Another operator might not want to allow a particular media type (e.g., video) at certain times of the day when the network is congested. Some of these policies are targeted at improving the QoS perceived by the users [46]. Other policies are intended to allow operators to implement a particular pricing model for their multimedia

services.

Since the use of proxy servers did not allow network operators to implement the policies above, these operators started deploying B2BUA (back-to-back user agents) that inspected and modified the session descriptions exchanged between user agents. These B2BUAs are usually referred to as SBCs.

While the use of SBCs allows operators to meet their requirements, they have some unfortunate architectural implications. In [57], we performed a survey among several SBC vendors. We documented how different functions (e.g., media management) were implemented and the architectural implications of implementing those functions in such ways. The fundamental implication of breaking end-to-end negotiations is that the session descriptions produced by the user agents need to be understood by nodes in the network. User agents cannot use any new extension in sessions between them until the network has been upgraded to support it as well. This, obviously, significantly slows down the pace at which new services and innovations can be deployed. SBCs sometimes also break end-to-end security. In those cases, user agents cannot distinguish between a network element trying to help them increase the QoS of their session and a man-in-the-middle launching an attack against them.

The initial answer from the research community to the issues above was to refer to SBCs as "evil" intermediaries and simply tell network operators not to deploy them because of their associated problems. SBCs at the SIP level were also compared with NATs at the IP level because both harm innovativeness. However, such answer ignored the economical incentives of network operators. Unsurprisingly, the rate at which SBCs got deployed kept increasing. In this, SBCs were also similar to NATs: the problems they caused were considered less important than the issues they resolved.

Kempf et al. [71] identify the lack of trust, and the appearance of new service and business models, as the most important trends opposing the end-to-end principle on the Internet [127] [33]. The protection of innovation is listed as one of the main desired properties of the end-to-end principle that needs to be kept on the Internet. Clark et al. [34] argue that new design strategies need to accommodate the growing tussle among and between different Internet players. Moors [86] argues that it is necessary to have information about a system in order to know how to apply the end-to-end principle to it. The need for notifications so that end users can detect the behavior of network intermediaries is considered important

when designing network services [40].

We propose to use the general principles we just described in order to implement policy control in SIP. In Publication I, we propose a mechanism, which is referred to as session policies, intended to meet the requirements of network operators without breaking the end-to-end nature of session negotiations between user agents. Session policies allow network operators to communicate their policies to user agents so that the user agents can take them into consideration when establishing sessions. Hilt et al. also described a framework for session policies in [60]. We worked together on specifying session policies for SIP [59]. We are not aware of other policy control mechanisms for SIP that share the characteristics we just discussed with the session policies approach.

There are two types of session policies: session-independent and session-specific policies. Session-independent policies apply to all sessions a user agent may establish. They are part of the configuration data the user agent gets from the network. Session-specific policies apply to a particular session. The user agents inform the network about the session they intend to establish and the network informs the user agents about the policies that are applicable to the session.

In Publication I, we also describe how to integrate session policies in the PCC (Policy and Charging Control) IMS architecture [1] [2]. We validated our results through a proof-of-concept prototype of the proposed architecture. Note, however, that session policies are a general mechanism also applicable to non-IMS scenarios [131].

We specified the details needed to implement session policies in SIP in [59], [58], and [61]. We chose to implement the communication channel between user agents and network elements providing policies using a SIP event package [58]. The use of an event package makes it possible for network elements to change their policies and inform the user agents during an ongoing session.

Figure 6.2 shows an example of a message flow that includes session policies. In this example, the originating user agent consults the policy server (1) before sending its initial INVITE request. The policy server provides the user agent with session-specific policies (3) that are applicable to the session that is about to be established. For example, the policy server may indicate that video streams are not allowed at that time of the day. The user agent takes this restriction into account when generating its offer (5). Once the user agent completes the offer/answer exchange with

the remote user agent, it informs the policy server about the resulting session (11). The policy server can provide additional policies in subsequent NOTIFY requests (13). For example, a few minutes later the policy server could inform the user agent that video streams are allowed.

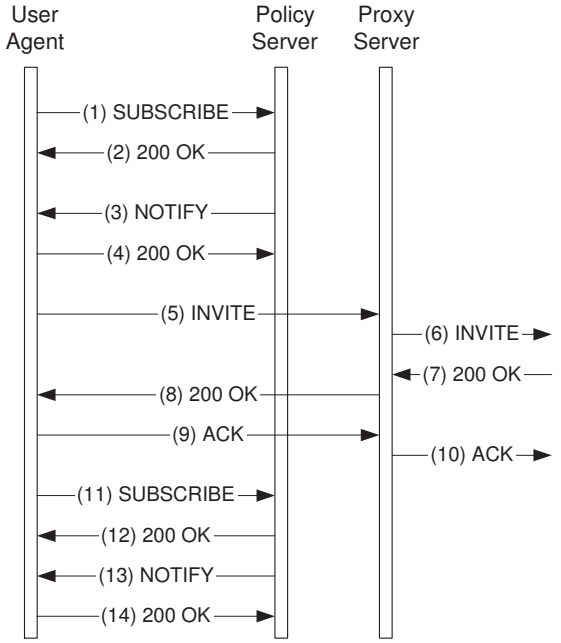

Figure 6.2. Message flow including session policies

6.3 Consent-based Communications

Unwanted traffic in SIP is specially problematic because of the nature of SIP applications. SIP applications are typically disruptive given that they alert its user about incoming communication attempts (e.g., the arrival of an instant message or an incoming call). A situation where users were continuously disrupted because of unwanted communication attempts would make it impossible for users to use those applications.

The design principles behind routing in a SIP network and routing in an IP network are similar. A sender generates a message or a packet that includes its intended destination and the network takes care of delivering the message or the packet to its destination. Networks operating in this way make it relatively easy and cheap for senders to send traffic to any recipient. This model does not help prevent unwanted traffic at all be-

cause the burden is placed on recipients to filter incoming traffic in order to discard the traffic they did not really want to receive. There are many examples of unwanted traffic at different layers. DoS attacks at the IP layer and SPAM in email applications are two well-known examples.

Handley et al. [51] propose a number of architectural changes in order to make the Internet more resistant to DoS attacks. One of the proposed changes is to only allow clients to contact other clients through a server (at least initially) and in the context of a particular service. Ballani et al. [7] propose an architecture where each host should explicitly declare to the network routing infrastructure what traffic it wants routed to it. Stoica et al. [137] propose an architecture that decouples the act of sending from the act of receiving. Anderson et al. [5] propose an architecture where instead of being able to send anything to anyone at any time, nodes must first obtain "permission to send" from the destination. Särelä et al. [128] propose an architecture that also moves the control from senders to receivers.

Paxon [100] analyzes different reflection attacks, some of which involve amplification (i.e., the reflector sends out a larger volume of traffic than the attacker sends to it). An important strategy to mitigate attacks is to avoid designing protocols or mechanisms that can return significantly larger responses than the size of the request, unless a handshake is performed to validate the client's source address [53]. In summary, applications should not be unknowingly used to cause such disruptions either (e.g., by avoiding becoming traffic amplifiers in a DoS attack launched by somebody else). It is important to make sure that SIP infrastructures meet these requirements.

Considering the general principles we have just described, we propose to have recipients give their consent before relaying communication attempts to them. In Publication II, we propose a mechanism to add consent-based communications to SIP. The mechanism addresses the unwanted traffic issue by allowing the implementation of white lists. Users only receive communication attempts from authorized users (i.e., users in their white lists). Even though users would still receive authorization requests from unauthorized users willing to be added to a white list, users do not need to process those requests in real time. Therefore, they are much less disruptive than incoming communication attempts. As discussed in [119], consent-based communications do not resolve all problems related to SPAM in SIP but they help with many of them. As discussed in Pub-

lication II, even though instant messaging users are used to closed-group communications, phone users are used to be able to call directly any other user even if they had not had any previous contact before. When being able to contact unknown users in this way is desirable, consent-based communications are not a good solution to handle their communications.

Our proposed consent mechanism is also designed to prevent SIP nodes to be used as traffic amplifiers by unauthorized users. SIP nodes do not perform traffic amplifications on behalf of a given user until the user has been authorized. The mechanism not only avoids traffic amplification related to communication attempts (e.g., forking an INVITE request to several destinations). It also avoids traffic amplification related to authorization requests (e.g., sending a large number of authorization requests).

We described the requirements for implementing consent-based communications in SIP in [117]. We specified the details needed to implement our proposed consent framework in [118], [15], and [16].

Figure 6.3 shows the architecture of our proposed consent framework. The concept of relays performing translations is central to our framework. A translation is an operation by which the URI a request is addressed to (which is referred to as the request's target URI) is replaced with one or more URIs (which are referred to as the request's recipient URIs). A SIP entity (e.g., a proxy or a B2BUA) performing a translation acts as a relay.

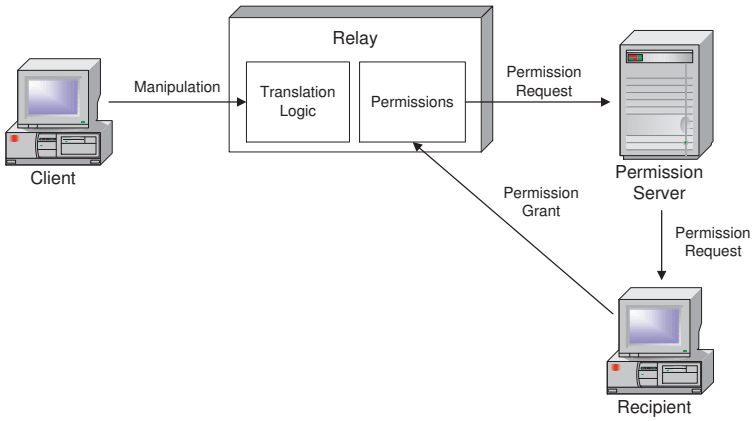

Figure 6.3. Consent Framework Architecture

A new recipient URI can be added to a given translation by manipulating the translation logic of a given relay. Such a manipulation is performed in different ways depending on the nature of the translation (e.g., by using REGISTER requests or by using XCAP). Our proposed consent

framework allows the relay to obtain authorization to send traffic to a newly added request URI.

In general, the relay will not have any means to identify the owner of the recipient URI being added to its translation logic. To resolve this issue, our proposed framework relies on the routing infrastructure to perform a return routability check. The relay simply sends an authorization request to the recipient URI and waits for an answer. As discussed in [90] and [89], return routability checks yield false positives if the routing infrastructure is compromised or if there is an attacker between the verifier and the address to be verified. Nevertheless, return routability checks are the best available mechanism in distributed scenarios where a given relay does not have a direct relationship with all the users in the system.

When sending authorization requests, relays need to avoid being used as traffic amplifiers. Our proposed framework uses a credit-based authorization mechanism to avoid that situation. As discussed in [141], [142], and [6], credit-based authorization can be used to limit the amount of data sent by a relay to an address for which the relay has not yet received authorization. Our proposed credit-based authorization, the relay is kept from sending more data to an unauthorized address than the amount of data the relay receives from the entity adding that unauthorized address to the relay's translation logic.

Relays need to keep authorization information for the recipient URIs that are part of their translation logic. This implies that a given relay may need to store authentication information for entities that do not have a direct relation with the relay (e.g., the owner of a recipient URI that has a different domain than the relay). While it is generally acceptable for relays to store simple authorization information (e.g., a given recipient is willing to receive SIP messages from a given sender), relays do not typically have incentives to store more complex authorization information (e.g., a given recipient is willing to only receive INVITE requests from a given sender only on weekends from 9 to 11 in the morning) and even less to perform filtering based on it. Therefore, our proposed document format for requesting consent [15] keeps the information stored by the relays at minimum.

Figure 6.4 shows an example of a message flow that includes the consent mechanism just described.

User A adds user B to the relay's translation logic (1). User A subscribes to the 'pending additions' event package at the relay (5) in order to be in-

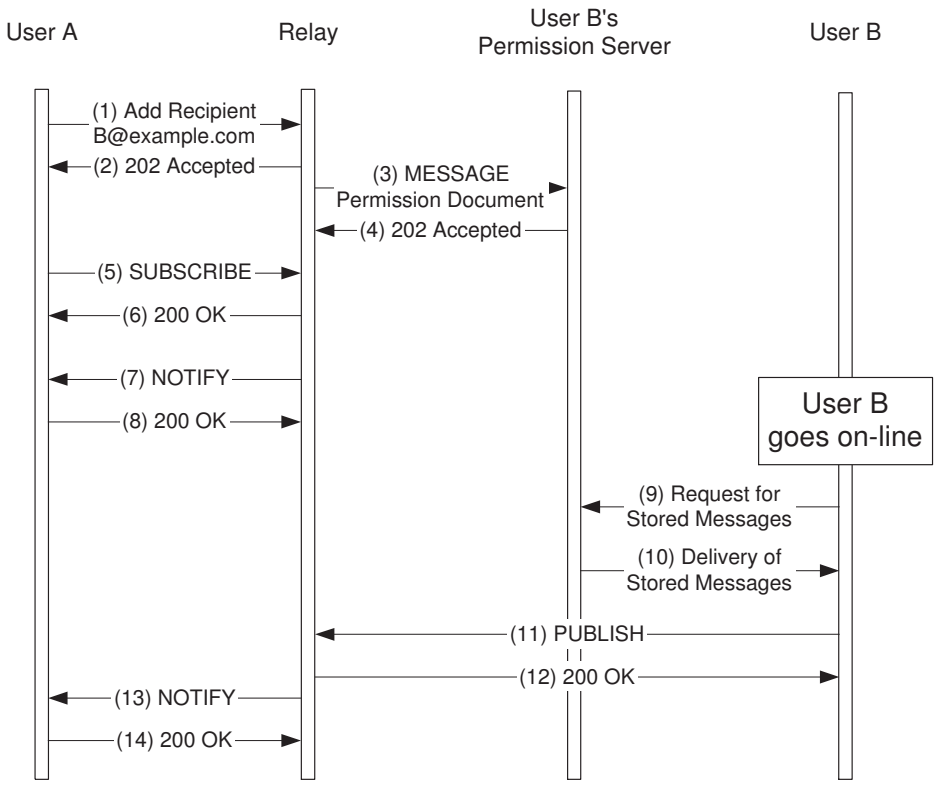

Figure 6.4. Consent Message Flow

formed of the permission-related status of B's addition. In order to ask B for permission to be added to the relay's translation logic, the relay sends a MESSAGE request (3) to B. Since user B is off-line when the MESSAGE request arrives to user B's permission server, the permission server stores it until user B goes on-line again. At this point, user B retrieves the contents of the MESSAGE request from the permission server (9). User B grants the relay permission to perform the translation described in the MESSAGE request by sending a PUBLISH request (11) to the relay. Finally, the relay informs user A that user B's URI has been added to the relay's translation logic using a NOTIFY (13).

7. Transport

Figure 4.1 in Chapter 4 shows the structure of the framework described in this thesis. The framework includes a transport function, which is discussed in this chapter. This chapter is based on our work on transporting SIP, which is documented in Publication III, Publication IV, and [123].

In Publication III, we compare SCTP, UDP, and TCP as transport protocols for SIP. Our evaluation is based on our implementation of SIP and SCTP in the network simulator (ns-2) [93]. We compare different aspects of the transport protocols (loss detection, congestion control, etc.) and analyze how they affect the transport of SIP signalling. Some of our simulations deal with HOLB (Head of the Line Blocking).

In Publication IV, we study the effects of HOLB on session establishment delays. Our study was based on experiments performed in a test bed and on the public Internet.

In [123], we specify how to transport SIP over SCTP. We also briefly describe the advantages of SCTP over TCP and UDP when transporting SIP.

Given that SIP was a signalling protocol whose messages required a reliable delivery, simply using TCP to transport SIP messages was arguably the most natural choice. Nevertheless, SIP was designed to be able to run on top of different transport protocols. In particular, the main SIP specification [124] described how to run SIP over TCP and UDP. The main idea behind being able to run SIP over UDP was to reduce session establishment times by avoiding the TCP three-way handshake. Reliability was achieved by implementing application-layer timeouts and retransmissions. SIP messages were assumed to be fairly small and never larger than the path MTU.

Experience gathered since SIP was originally designed has shown that the decision to allow UDP to be a valid transport for SIP had also impor-

tant disadvantages. In retrospect, its disadvantages are actually arguably more important than its advantages.

SIP messages in current deployments are often much larger than the path MTU. When transfered over UDP, such messages cause IP-level fragmentation. Even if an entity chose to only send small requests over UDP (and send larger requests over TCP instead), the responses to those small requests may be much larger than the requests themselves. Given that a request and a response need to use the same transport protocol, those large responses to small requests would still cause IP-level fragmentation. Additionally, the original SIP specification [54] allowed user agents to implement either TCP or UDP. Consequently, there are legacy user agents that do not support TCP at all, even if the current SIP specification [124] mandates all SIP entities to support both TCP and UDP.

IP-level fragmentation causes a number of problems. Even if only a single fragment of a UDP packet is lost, the whole UDP packet will be retransmitted. This can be quite inefficient because fragments that had been successfully transmitted are retransmitted anyway. Additionally, the fact that only the first fragment carries the UDP header makes NAT and firewall traversal impossible in some situations (e.g., a NAT or firewall that refuses to keep state when receiving out of order fragments in order to offer better DoS protection).

Supporting UDP also adds complexity to SIP applications. Implementing application-layer timeouts and transmissions, and the handling of multiple transport protocols comes with a cost in terms of complexity.

HOLB occurs when the data associated with multiple logical sessions is interleaved over a single TCP connection. If a segment for one of the session gets lost, the delivery of messages for the other sessions has to wait until the lost segment has been retransmitted and correctly received. Thus, a logical session blocks because of loss in another, unrelated one.

When used between proxy servers, UDP avoids the HOLB problem (HOLB appears when two proxies use the same TCP connection to carry multiple SIP dialogs between them). However, UDP does not provide any congestion control. Therefore, such proxy servers may overload the network to the point of collapse in periods when they need to handle heavy loads.

Therefore, both UDP and TCP have advantages and disadvantages when transporting SIP messages. Additionally, we specified how to transport SIP over SCTP [123] so that it was possible to study whether SCTP could be a better transport for SIP than UDP or TCP in certain scenarios.

There have been a few studies on the effects of HOLB. Grinnem et. al. [47] studied the increase in the average one-way delay introduced by HOLB in SCTP. The study was based on experiments in an emulated network environment including competing TCP traffic. Scharf et. al. [129] analyzed the increase in the average two-way delay (i.e., response time in a request-response based protocol) introduced by HOLB in SCTP. The study was based on experiments on an emulated network environment including random packet losses.

In Publications III and IV, we compare UDP, TCP, and SCTP for transporting SIP traffic. In particular, we analyze the effects of HOLB. Our results have a more general applicability than previous studies because our simulations and experiments included scenarios with different packet-loss patterns, call rates, link delays, and routers' queue sizes. We analyze the single-session and multiple-session scenarios and discuss what is the best transport protocol for each of them.

The single-session scenario consists of a user agent exchanging SIP messages with its proxy server. All the SIP messages exchanged generally correspond to a single session. Consequently, HOLB avoidance, one of the biggest advantages of SCTP, is not typically an issue in this scenario. As described in Publication III, SCTP has a few additional advantages over TCP (e.g., message orientation and multihoming). However, most existing NATs do not understand SCTP and, thus, filter SCTP messages out. This is a fundamental disadvantage of SCTP with respect to TCP. Therefore, we recommend to use TCP in this type of scenario.

The multiple-session scenario consists of two proxy servers exchanging SIP messages between them over a single transport connection. The SIP messages exchanged typically correspond to many different sessions between different user agents. Consequently, HOLB is a an important issue in this scenario. Figure 7.1 illustrates how HOLB affects the delivery of SIP messages corresponding to different sessions. The X-axis represents the time when a particular SIP message is passed to the transport layer by the sender (in ms) and the Y-axis represents when that message is delivered to the receiving application by the transport layer (in ms).

Figure 7.1 shows the behaviour of a TCP connection and an SCTP association under the same traffic. While the SCTP association avoids HOLB, the TCP connection does not avoid it.

The packet that was sent at t=60882 ms gets lost. The sender retransmits this packet and this retransmission finally arrives at t=61193 ms.

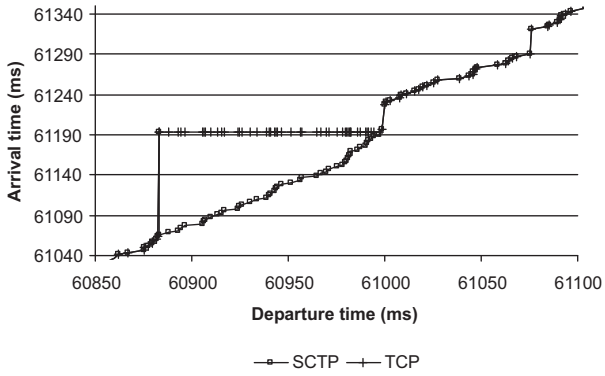

Figure 7.1. HOLB Effects

Before this message is finally received, the receiver has been receiving new DATA chunks containing new SIP messages from the sender.

The graph shows that while SCTP delivers all the SIP messages received regardless of the packet lost (bottom line in the graph), TCP buffers all this data without passing it to the application (top line). When the retransmission of the packet lost is finally received, TCP passes all the data at once to the application layer. This sudden delivery of several SIP messages is represented by the flat line in the graph. The delay introduced by HOLB to a particular packet is the difference between the two lines. When a packet is not affected by HOL blocking both lines coincide.

As discussed in Publications III and IV, depending on the network conditions at any given time, HOLB can easily become a significant factor in the session establishment delay. Therefore, we recommend to use SCTP instead of TCP in this type of scenario in order to avoid HOLB.

8. Multihoming, Mobility, Security, and NAT Traversal

Figure 4.1 in Chapter 4 shows the structure of the framework described in this thesis. The framework includes support functions related to mobility, multihoming, security, and NAT traversal. All these functions are discussed in this section.

This section is based on our work on increasing SCTP's security, which is documented in Publication V, and [135], our work on combining HIP and SIP, which is documented in Publication VI, and our work on traffic-flow management in HIP, which is documented in Publication VII.

In Publication V, we studied the effect of dynamic multiaddressing (i.e., mobility and multihoming) in SCTP's security. We identified vulnerabilities and ways to fix them. In [135], we described how the SCTP specifications were modified in order to address those vulnerabilities.

In Publication VI, we proposed a framework to combine SIP and HIP. We describe the advantages derived from using HIP to implement some functions in a SIP environment. Additionally, we evaluate the performance implications of using HIP in a SIP environment using a proof-of-concept prototype implementation.

In Publication VII, we proposed a mechanism to allow two HIP endpoints to choose different paths for different flows between them. Our mechanism is based on ICE (Interactive Connectivity Establishment) [116] and the SIMA (Simultaneous Multiaccess) HIP extension [101]. Our proposed mechanism gets the information about the available paths from ICE, and the preferences from the user or application as input. With all this information, the system decides which path should be used for which flow and routes all flows accordingly. When network conditions change, flows can be rerouted if needed. We evaluated our proposed mechanism with a proof-of-concept prototype implementation.

8.1 Dynamic Multiaddressing in SCTP

SCTP [134] provides mobility and multihoming functionality. SCTP endpoints can manage (e.g., add and remove) the IP addresses at which they are reachable during an association [136]. SCTP was originally designed to transport telephony signalling. Some of the assumptions made during SCTP's design were valid in that context. However, when SCTP is used as a general-purpose transport protocol, some of those design assumptions do not hold any longer. As a consequence, some threats against SCTP are more likely to be exploited by attackers.

Part of the security research on SCTP has focused on studying different end-to-end security solutions such as TLS [69] and IPsec [8]. Lindskog et al. [79] compare different end-to-end security solutions for SCTP and propose an alternative solution. Unurkhaan et al. [139] also propose a security extension to SCTP. Nordhoff [92] compares the performance of different end-to-end security solutions. Nordhoff also analyzes different DoS attacks against SCTP.

Our work focused on the SCTP mechanisms that could be abused to launch attacks against the protocol. Based on our work, the SCTP specifications were modified in order to mitigate the attacks we described. In Publication V, we described the assumptions that were invalidated when SCTP was used as a general-purpose transport protocol and discussed how to mount attacks that took advantage of them. These attacks included DoS, connection hijacking, and packet flooding attacks. We verified that our attacks were indeed possible and realistic by experimenting with three open-source SCTP implementations. In Publication V, we also presented potential solutions against the attacks. In [135], we described how the SCTP specifications were modified in order to address those vulnerabilities.

8.2 Combining HIP and SIP

Mobility, multihoming, security, and NAT traversal are all important features in a SIP system. Nikander et al. [91] propose to use HIP to provide integrated mobility, multihoming, and security.

HIP [87] proposes a new layer in the TCP/IP protocol stack. This new layer is located below the transport layer, as shown in Figure 8.1, and con-

sists of cryptographically generated host identifiers. HIP provides multihoming, mobility, security, and NAT traversal functionality.

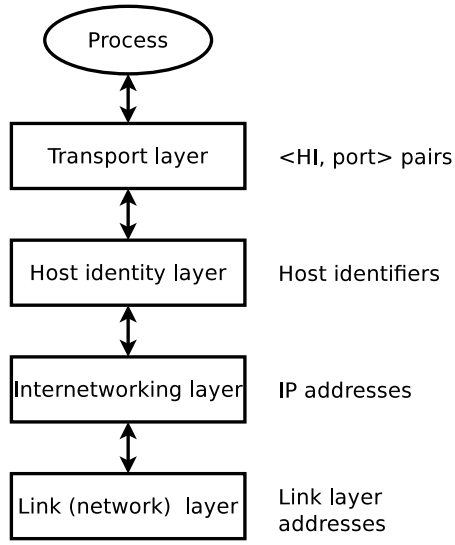

Figure 8.1. HIP in the TCP/IP stack

HIP implements a so called ID/locator split. Instead of using the IP address as the identifier for the host, with HIP, the identifier is the public key of an asymmetric cryptographic key pair. Also, instead of binding connections to the IP address, transport layer protocols can now bind to a presentation of the host identifier: Host Identity Tag (HIT). The host identifier can remain the same regardless of the IP address that is currently used for it. Because the binding between the host identity and the IP address is not fixed, the IP address can change without breaking the transport layer connections.

With this solution, mobility and multi-homing can be handled in a more natural way since multiple IP address can be bound to a host identifier and the underlying IP address can be changed dynamically. Also, because the host identifier is an asymmetric cryptographic key, the host can prove that it is allowed to use the identifier with the private key. This solves the problem with malicious hosts trying to change the destination of the traffic. IPsec Encapsulating Security Payload (ESP) [87] provides confidentiality and integrity protection for all the traffic that is exchanged between the hosts.

In Publication VI, we propose to combine HIP and SIP to provide multihoming, host mobility, hop-by-hop security, protection against flooding attacks, and NAT traversal functionality.

SIP and HIP provide complementary types of mobility management. We propose to use SIP to provide user and session mobility and to use HIP to provide host mobility. User mobility provides a user with the ability to be reachable under the same identifier regardless of the user's location. Session mobility involves the transfer of a ongoing session between different devices. Host mobility occurs when a host changes its IP address.

Connections managed by HIP are started with a four-way handshake referred to as the HIP base exchange. When used in a SIP environment, the HIP base exchange protects SIP nodes from being used to launch flooding attacks such as the well-known media hammering attack or attacks based on third party registrations. The HIP base exchange allows SIP nodes to make sure they only send media to the correct parties (and not to victims).

HIP also provides hop-by-hop security, which is based on IPsec. In a SIP environment, HIP-based hop-by-hop security can replace TLS to protect SIP signalling and other media security mechanisms to protect media traffic. Being able to reuse the same security association for both SIP and media traffic reduces the session establishment delay in some cases (e.g., when SIP is used in an overlay, as discussed in Chapter 9).

HIP includes a NAT traversal mechanism based on ICE. As discussed above, the same security association can carry SIP and media traffic. Therefore, it is only necessary to run the ICE procedures once to establish a security association, instead of running them several times for SIP and for different media streams. Being able to run ICE procedures only once for both SIP and media traffic reduces the session establishment delay in some cases (e.g., when SIP is used in an overlay, as discussed in Chapter 9).

8.3 Flow Management in HIP

As discussed in Section 8.2, one of the advantages of using HIP is that a single security association can carry all traffic between two endpoints. The endpoints use ICE to find a working path between them and establish a security association over that path. Keeping the endpoints from re-running ICE and security establishment procedures for each new flow being established between the endpoints saves both time and bandwidth. However, in some circumstances, different flows need to be treated differently in the network.

HIP's original design did not include NAT traversal capabilities but it was later extended to include an ICE-based NAT traversal mechanism [75]. HIP's original design only supported basic multihoming capabilities. Pierrel et al. [101] proposed to provide HIP with more advanced multihoming capabilities by allowing HIP hosts to use multiple accesses simultaneously.

Our work is based on the two HIP extensions we just described. In Publication VII, we proposed a mechanism to allow two HIP endpoints to choose different paths for different flows between them. Our proposed mechanism allows users and applications to specify policies for traffic flows. Given a policy for a particular flow, HIP endpoints automatically choose the most appropriate path for the flow at any given time. The ICE procedures used to find connectivity and keep NAT bindings alive are reused to measure the characteristics of the working paths between the endpoints. In this way, the system can react when the characteristics of a particular path change.

Figure 8.2 shows the architecture we propose to implement flow management in HIP. The architecture consists of three modules: the policy module, the ICE module, and the SIMA module.

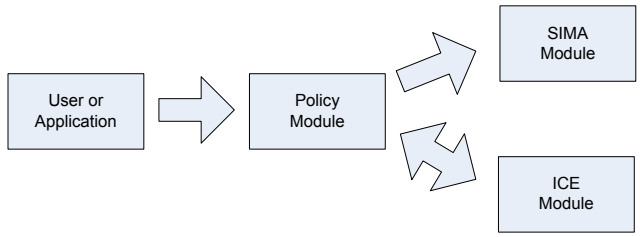

Figure 8.2. Proposed architecture for flow management in HIP

The policy module gets input from the user or application on the preferences for different traffic flows. The policy module obtains further information about different available paths from the ICE module. The ICE module measures path features such as currently available bandwidth and delay and informs the policy module. Depending on the preferences given to the policy module, the policy module instructs the ICE module to measure certain features. For example, if the policy module has not gotten any bandwidth-related preferences, it will instruct the ICE module not to do any bandwidth measurements. With all the information from the ICE module and the preferences from the user or application, the policy module decides which path should be used for which flow. The policy module,

then, instructs the SIMA module to route traffic flows accordingly.

This functionality is useful in cases where two hosts exchange different types of traffic between them and want those different types of traffic to be treated differently. For example, the hosts may want to use a high-bandwidth path for streaming video and a low-latency path for signalling traffic.

9. Overlays

Figure 4.1 in Chapter 4 shows the structure of the framework described in this thesis. The framework includes functions related to overlays. These functions are discussed in this chapter.

This chapter is based on our work on building HIP-based overlays, which is documented in Publication VI, Publication VIII, [25], [73], [23], and [22].

In Publication VI, we propose to build P2PSIP (Peer-to-peer Session Initiation Protocol) overlays (i.e., overlays that route SIP traffic) based on HIP. In Publication VIII, we describe an architecture to build such overlays where HIP is used to perform connection management while other functions such as data storage and retrieval, or overlay maintenance, are implemented using a different protocol. Also in Publication VIII, we evaluate the performance gains related to NAT traversal derived from the use of this architecture. In [25], we specify in further detail our proposed architecture to build HIP BONE (HIP Based Overlay Networking Environment) overlays. In [73], we specify the details needed to implement a HIP BONE overlay using RELOAD (REsource LOcation And Discovery). In [23], we specify how to transfer HIP-encapsulated data between two endpoints that have not performed a HIP based exchange. In [22], we define the extensions needed to implement multi-hop and source routing in HIP.

There are scenarios where even though no communication infrastructure is available (e.g., disaster areas), there is a need to communicate (e.g., among the members of a rescue team). An overlay is a network that is built on top of another network (e.g., a peer-to-peer overlay that runs on top of the Internet). Overlay networks are an excellent choice to enable communications in these types of scenarios. Overlays enable the construction of different types of peer-to-peer systems where functions typically

provided by the communication infrastructure (e.g., rendezvous) are provided by the peers instead. In [20], we analyzed the existing peer-to-peer architectures.

Peer-to-peer communication systems mostly use overlays based on DHTs (Distributed Hash Tables). There exist several DTH algorithms: Stoica et. al. proposed Chord [138], Rowstron et. al. proposed Pastry [126], Ratnasamy et. al, proposed [109], and Zhao et. al, proposed Tapestry [143]. In [55], we evaluated different DHT algorithms from the viewpoint of interpersonal communications. In [56], we proposed a method to decentralize applications that were originally centralized.

RELOAD (REsource LOcation And Discovery) [68] is a protocol that allows to build overlay networks using different algorithms. In [81] and [83], we studied how an overlay could estimate its operating conditions and auto-configure its parameters depending on the current conditions. In [80], we studied the performance of a Chord-based overlay under different network conditions. In [82], we specify a service discovery mechanism for RELOAD.

Gurtov et. al. [48] proposed a network architecture based on HIP in [137] where an overlay provides rendezvous services for HIP nodes.

In Publication VIII, we propose an architecture to build HIP-based overlays. Overlays typically require three types of operations: overlay maintenance, data storage and retrieval, and connection management. Overlay maintenance operations deal with nodes joining and leaving the overlay and with the maintenance of the overlay's routing tables. Data storage and retrieval operations deal with nodes storing, retrieving, and removing information in or from the overlay. Connection management operations deal with the establishment of connections and the exchange of lightweight messages among the nodes of the overlay, potentially in the presence of NATs.

There are peer protocols (e.g., RELOAD [68]) that provide all these three functions. In our architecture, we propose to use HIP to perform connection management. Data storage and retrieval, and overlay maintenance are still implemented using a peer protocol (i.e., a protocol different than HIP).

Figure 9.1 shows our architecture, as described in Publication VIII. All protocol messages are carried either on top of a data connection previously established by HIP or on top of a HIP message (which can be, in turn, transported over a data connection or directly over IP).

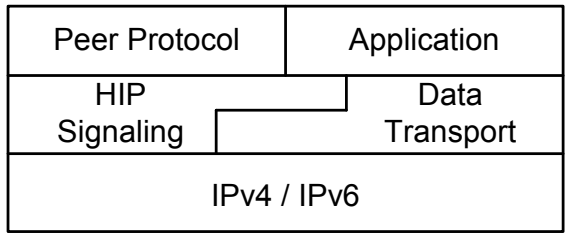

Figure 9.1. Layers in the HIP BONE architecture

The HIP base exchange is used to establish data connections between HIP nodes also in overlay scenarios. In non-overlay scenarios, a node sends an I1 message (the first message of the HIP base exchange) to a remote node through the remote node's rendezvous server. However, in overlay scenarios, the rendezvous service is provided by the overlay. Therefore, the routing tables of the overlay nodes determine how the I1 is routed from its sender to its destination nodes.

Different overlays use different types of routing. Routing methods can be classified as recursive or iterative. In recursive routing, a message (e.g., an I1 message) traverses several nodes in the overlay until it reaches its destination without returning to its sender. In iterative routing, each overlay node handling the message returns it to its sender with a suggestion on which node the sender needs to send the message to next.

Recursive routing can be divided into forward-only and symmetric routing. In forward-only routing, the response to a message is routed independently to the original message. The routing of the response is based on the overlay's routing tables. In symmetric routing, the response follows the same path (in reverse order) the original message followed to reach its destination. We had to extend HIP [22] in order to support symmetric recursive routing.

Overlay nodes willing to exchange application-layer messages can establish a direct data connection between them. However, establishing a data connection in presence of NATs can be a time consuming process. For short-lived transactions (e.g., a query-response transaction), routing the application-layer messages over the overlay instead of over a direct connection is more efficient. We extended HIP [23] so that it can transport application-layer data between overlay nodes. We also use the HIP extension [72] that allows sending HIP messages over already-established data connections.

Our architecture where HIP (as opposed to a peer protocol) is used to

perform connection management in an overlay has a set of advantages. Our architecture can be used by any peer protocol. This keeps each peer protocol from defining primitives needed for connection management (e.g., primitives to establish connections and to tunnel messages through the overlay) and NAT traversal. Having this functionality at a lower layer allows multiple upper-layer protocols to take advantage of it.

Additionally, having a solution that integrates mobility and multihoming is useful in many scenarios. Peer protocols do not typically specify mobility and multihoming solutions. Combining a peer protocol including NAT traversal with a separate mobility mechanism and a separate multihoming mechanism can easily lead to unexpected interactions.

In our architecture, HIP can manage all the data connections between two nodes in an integrated manner. Our experiments with P2PSIP overlays using our architecture show that this integrated management results in a significant reduction in the session establishment delay in the presence of NATs, as described in Publication VIII. Nevertheless, if needed, HIP can also manage different connections with different requirements in different ways, as described in Publication VII.

10. Conclusions

We have described our proposed framework and our contributions in the areas it covers. When describing the different parts of the framework, we have also discussed research work related and relevant to each area of the framework. In Chapter 2, we presented the main goal of this thesis. We aimed to build a framework based on SIP to enable communication services. The framework was intended to have four fundamental properties. It needed to be available, secure, high performing, and oriented to innovations. In this chapter, we discuss each of these properties in the context of our proposed framework.

As discussed in Section 5.1, SIP provides most of the primitives commonly used by applications handling one-to-one sessions. Additionally, as discussed in Section 5.2 the framework includes primitives to efficiently manage multiparty sessions. These primitives are used to access URI-list services. As discussed in Section 5.3, the framework also includes primitives to manage sessions that involve floor control.

The framework is available in many ways. In the framework, mobility, multihoming, and NAT traversal are handled by HIP, as described in Section 8.2. The mobility support allows users to access services even when they are on the move. The multihoming support allows users to access services even when an access becomes unavailable. In environments where HIP is not used, SCTP can also provide mobility and multihoming, as discussed in Section 8.1. The ICE-based NAT traversal capabilities provided by HIP makes the framework available in environments with NATs. Early Internet communication services did not work in this type of environments and, nowadays, being able to traverse NATs is considered an essential feature. NAT traversal makes the framework available to a high number of users whose connections are behind NATs.

As described in Section 6.1, the framework is accessible from the PSTN.

Being able to interwork with the PSTN substantially increases the number of users for whom the services provided are made available, even though PSTN users may be able to access the framework in a limited way. The framework can also be made available in scenarios lacking an infrastructure of SIP servers, as described in Chapter 9.

DoS attacks are attacks against the availability of a service. As discussed in Section 6.3, consent-based communications provide protection against DoS attacks. Additionally, the HIP's cryptographic puzzles and SCTP's stateless return routability tests, both of which are performed at connection establishment time, protect against resource exhaustion attacks.

In terms of security, as discussed in Section 8.2, HIP provides confidentiality and integrity protection to both signalling and media. While HIP security is currently based on IPsec, future extensions may make it possible to provide other types of protection.

In order to achieve high performance, we needed to integrate the operations in the framework in an efficient way. As discussed in Chapter 9, the integrated connection management employed in the framework results in a significant reduction in the session establishment delay in the presence of NATs in some scenarios. Nevertheless, if needed, connections with different requirements can be managed in different ways, as discussed in Section 8.3. Minimizing transport-related delays also helps increase the performance of a system. As discussed in Chapter 7, avoiding HOLB can significantly reduce session establishment delays.

With respect to innovativeness, as discussed in Section 6.2, session policies allow network operators to communicate their policies to the endpoints without breaking end-to-end session negotiations between them. End-to-end negotiations enable the introduction of new and innovative services at a much faster rate because intermediaries do not need to be updated every time a new service is to be provided. Additionally, the use of overlays also facilitates the introduction of new services by removing the need to wait for infrastructures to be deployed before the actual introduction of the services. Nevertheless, the innovativeness of a framework is effectively decided on the markets. The actual introduction of new and innovative services based on our framework will be the ultimate criterion of how well we have succeeded in this respect.

10.1 Future Work

The protocol work in this dissertation follows a model that has been (and will continue being) widely used on the Internet and other networks such as the PSTN. In this model, entities communicate with each other using agreed upon (e.g., standardized) protocols. There are typically several implementations of the same protocol, using different programming languages or running on different platforms. Each entity is free to choose any of those implementations as long as the implementation is compliant with the protocol definition.

In the case of SIP, the same protocol is used end-to-end. That is, both user agents and network entities (e.g., proxy servers) use SIP. Note that this is not the case in other networks such as the PSTN where different protocols are used on the user-to-network interface and on the network-to-network interface.

SIP allows the implementation of new features in several ways. Some types of features require the design and standardization of new SIP extensions, which can be a time-consuming process. Additionally, users typically need to install a new application in order to access SIP-based services. Installing applications can be problematic in some scenarios. For example, a user may not trust the application to give it full access rights to its terminal's resources, a user may not be able to install applications on a terminal that is not his or her own, or the application may not be available for the user's terminal platform.

There is currently a significant amount of work trying to overcome some of these issues. On the web, the mobile code model is widely used nowadays. The client requests a service from a web server and the web server delivers a piece of code to the client. The client executes that piece of code as part of getting the requested service. There is work aimed to adapt this model to real-time communications as well.

In the mobile code model, the need for standard protocols changes. Instead of designing protocols such as SIP, which include many functional elements, the mobile code model requires a standard protocol for delivering the mobile code and a standard programming language. On the web, the standard delivery protocol is HTTP and the standard programming language is javascript. Also, the execution environment in the mobile code model is not the operating system but the browser instead.

Even though there are real-time communications services that use the

mobile code model available on the Internet, they use proprietary browser plugins. There is ongoing work to make it possible to use the mobile code model for real-time communications services in a standard way in all browser platforms.

The results of this work remain to be seen. However, it will likely have an impact in some of the services that will be provided in the future. How it will exactly relate to the work in this dissertation and to SIP is also still unclear. A likely scenario is that some services continue to use SIP in an end-to-end fashion while others use the mobile code model from a browser to a server and SIP from the server on.

Even though there are still many open issues regarding the mobile code model (e.g., related to customer lock in), the model has significant potential to enable innovations. Future research will be needed in order to further study this model and its applicability to more general scenarios (e.g., involving a federation of service providers).

This dissertation considers P2P architectures that allow distributing the processing within a communication network. Hybrid architectures can still include servers but distribute part of their processing to endpoints or other servers participating in a P2P overlay. In the last years, cloud computing technologies have emerged as an alternative to distribute the processing in a system. In the future, there will probably be deployments following all of these architectural approaches.

Research on so called Future Internet architectures is currently getting significant attention. In particular, there is a lot of research on information-centric networking architectures. Some of the ideas related to HIP-based overlays presented in this dissertation are relevant in that area (e.g., implementing NAT traversal and flow management at the ID/locator split level).

During the last years there has been a shift in the context in which communications take place. More and more communications take place in the context of social networks nowadays. The ideas related to consent-based communications presented in this dissertation are relevant in that area. In a social network, the management of which users are allowed to initiate which types of communications with other users is generally considered an essential feature.

Bibliography

[1] 3GPP. 2008. Policy and Charging Control over Gx Reference Point. TS 29.212, 3rd Generation Partnership Project (3GPP).

[2] 3GPP. 2008. Policy and Charging Control over Rx Reference Point. TS 29.214, 3rd Generation Partnership Project (3GPP).

[3] 3GPP. 2009. Open Service Access (OSA) Application Programming Interface (API); Part 1: Overview. TS 29.198-1, 3rd Generation Partnership Project (3GPP).

[4] 3GPP. 2011. General Packet Radio Service (GPRS); Service description; Stage 2. TS 23.060, 3rd Generation Partnership Project (3GPP).

[5] T. Anderson, T. Roscoe, and D. Wetherall. 2003. Preventing Internet Denial-of-service with Capabilities. ACM SIGCOMM Computer Communnication Review (CCR) 34, no. 1, pages 39–44.

[6] J. Arkko, C. Vogt, and W. Haddad. 2007. Enhanced Route Optimization for Mobile IPv6. RFC 4866.

[7] Hitesh Ballani, Yatin Chawathe, Sylvia Ratnasamy, Timothy Roscoe, and Scott Shenker. 2005. Off by Default. In: Proceedings of the 4th ACM Workshop on Hot Topics in Networks (HotNets, pages 1–6.

[8] S. Bellovin, J. Ioannidis, A. Keromytis, and R. Stewart. 2003. On the Use of Stream Control Transmission Protocol (SCTP) with IPsec. RFC 3554.

[9] J. Bound, Y. Pouffary, S. Klynsma, T. Chown, and D. Green. 2007. IPv6 Enterprise Network Analysis - IP Layer 3 Focus. RFC 4852.

[10] G. Camarillo. 1998. IP Telephony Gateways. M.Sc. thesis. Royal Institute of Technology, Sweden.

[11] G. Camarillo. 2001. SIP Demystified. McGraw- Hill.

[12] G. Camarillo. 2003. Compressing the Session Initiation Protocol (SIP). RFC 3486.

[13] G. Camarillo. 2006. Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams. RFC 4583.

[14] G. Camarillo. 2007. Connection Establishment in the Binary Floor Control Protocol (BFCP). RFC 5018.

[15] G. Camarillo. 2008. A Document Format for Requesting Consent. RFC 5361.

[16] G. Camarillo. 2008. The Session Initiation Protocol (SIP) Pending Additions Event Package. RFC 5362.

[17] G. Camarillo. 2009. Message Body Handling in the Session Initiation Protocol (SIP). RFC 5621.

[18] G. Camarillo and M. Garcia-Martin. 2008. The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds, 3rd Edition. Wiley.

[19] G. Camarillo, C. Holmberg, and Y. Gao. 2011. Re-INVITE and Target-Refresh Request Handling in the Session Initiation Protocol (SIP). RFC 6141.

[20] G. Camarillo and IAB. 2009. Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability. RFC 5694.

[21] G. Camarillo and A. Johnston. 2008. Conference Establishment Using Request-Contained Lists in the Session Initiation Protocol (SIP). RFC 5366.

[22] G. Camarillo and A. Keranen. 2010. Host Identity Protocol (HIP) Multi-Hop Routing Extension. RFC 6028.

[23] G. Camarillo and J. Melen. 2011. Host Identity Protocol (HIP) Immediate Carriage and Conveyance of Upper-Layer Protocol Signaling (HICCUPS). RFC 6078.

[24] G. Camarillo, A. Niemi, M. Isomaki, M. Garcia-Martin, and H. Khartabil. 2008. Referring to Multiple Resources in the Session Initiation Protocol (SIP). RFC 5368.

[25] G. Camarillo, P. Nikander, J. Hautakorpi, A. Keranen, and A. Johnston. 2011. HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment (BONE). RFC 6079.

[26] G. Camarillo, J. Ott, and K. Drage. 2006. The Binary Floor Control Protocol (BFCP). RFC 4582.

[27] G. Camarillo, A. B. Roach, J. Peterson, and L. Ong. 2002. Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping. RFC 3398.

[28] G. Camarillo, A. B. Roach, J. Peterson, and L. Ong. 2003. Mapping of Integrated Services Digital Network (ISDN) User Part (ISUP) Overlap Signalling to the Session Initiation Protocol (SIP). RFC 3578.

[29] G. Camarillo and A.B. Roach. 2008. Framework and Security Considerations for Session Initiation Protocol (SIP) URI-List Services. RFC 5363.

[30] G. Camarillo, A.B. Roach, and O. Levin. 2008. Subscriptions to Request-Contained Resource Lists in the Session Initiation Protocol (SIP). RFC 5367.

[31] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. 2002. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428.

[32] C. Christensen. 2000. The Innovator's Dilemma. HarperCollins Publishers.

[33] D. Clark. 1988. The Design Philosophy of the DARPA Internet Protocols. In: Proceedings of ACM SIGCOMM: Symposium Proceedings on Communications Architectures and Protocols, pages 106–114.

[34] D. Clark, J. Wroclawski, K. Sollins, and R. Braden. 2005. Tussle in cyberspace: defining tomorrow's internet. IEEE/ACM Transactions on Networking 13, no. 3, pages 462–475.

[35] B. Clayton. 2007. Securing Media Streams in an Asterisk-based Environment and Evaluating the Resulting Performance Cost. M.Sc. thesis. Rhodes University, South Africa.

[36] F. Depaoli and F. Tisato. 1991. Coordinator: a Basic Building Block for Multimedia Conferencing Systems. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM), pages 2049–2053.

[37] T. Dierks and E. Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246.

[38] H. Dommel and J. J. Garcia luna aceves. 1995. Floor control for Activity Coordination in Networked Multimedia Applications. In: Proceedings of 2nd Asian-pacific Conference on Communications (APPC), pages 1–5.

[39] S. Donovan. 2000. The SIP INFO Method. RFC 2976.

[40] B. Fenner. 2002. IANA Considerations for IPv4 Internet Group Management Protocol (IGMP). RFC 3228.

[41] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. 1999. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616.

[42] N. Freed and N. Borenstein. 1996. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045.

[43] M. Garcia-Martin and G. Camarillo. 2008. Extensible Markup Language (XML) Format Extension for Representing Copy Control Attributes in Resource Lists. RFC 5364.

[44] M. Garcia-Martin and G. Camarillo. 2008. Multiple-Recipient MESSAGE Requests in the Session Initiation Protocol (SIP). RFC 5365.

[45] F. Gens, R. Mahowald, R. Villars, J. Gantz, H. Morris, S. Hendrick, M. Turner, R. Shuchat, M. Ballou, M. Eastwood, C. Morris, S. Matsumoto, C. Ingle, D. Bradshaw, L. Fernandez, N. Wallis, and V. Kroa. 2010. Worldwide and Regional Public IT Cloud Services 2010-2014 Forecast. IDC , no. 223549.

[46] R. Good and N. Ventura. 2008. Linking Session Based Services and Transport Layer Resources in the IP Multimedia Subsystem. In: Proceedings of South Africa Telecommunication Networks and Applications Conference (SATNAC), pages 1–6.

[47] K.J. Grinnem, T. Andersson, and A. Brunstrom. 2005. Performance Benefits of Avoiding Head-of-Line Blocking in SCTP. In: Proceedings of the Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS), page 44.

[48] A. Gurtov, D. Korzun, A. Lukyanenko, and P. Nikander. 2008. Hi3: An Efficient and Secure Networking Architecture for Mobile Hosts. Computer Communications 31, no. 10, pages 2456–2467.

[49] M. Handley. 1997. On Scalable Internet Multimedia Conferencing Systems. Ph.D. thesis, University of London, UK.

[50] M. Handley, J. Crowcroft, C. Bormann, and J. Ott. 1999. Very Large Conferences on the Internet: the Internet Multimedia Conferencing Architecture. Computer Networks 31, no. 3, pages 191–204.

[51] M. Handley and A. Greenhalgh. 2004. Steps Towards a DoS-resistant Internet Architecture. In: Proceedings of ACM SIGCOMM: Workshop on Future Directions in Network Architecture, pages 49–56.

[52] M. Handley, V. Jacobson, and C. Perkins. 2006. SDP: Session Description Protocol. RFC 4566.

[53] M. Handley, E. Rescorla, and IAB. 2006. Internet Denial-of-Service Considerations. RFC 4732.

[54] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. 1999. SIP: Session Initiation Protocol. RFC 2543.

[55] J. Hautakorpi and G. Camarillo. 2007. Evaluation of DHTs from the Viewpoint of Interpersonal Communications. In: Proceedings of the 6th ACM International Conference on Mobile and Ubiquitous Multimedia (MUM), pages 74–83.

[56] J. Hautakorpi, G. Camarillo, and D. Lopez. 2009. Framework for Decentralizing Legacy Applications. In: Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), pages 544–549.

[57] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, and M. Bhatia. 2010. Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments. RFC 5853.

[58] V. Hilt and G. Camarillo. 2010. A Session Initiation Protocol (SIP) Event Package for Session-Specific Session Policies. Internet-Draft draft-ietf-sipping-policy-package-08, Internet Engineering Task Force. Work in progress.

[59] V. Hilt, G. Camarillo, and J. Rosenberg. 2011. A Framework for Session Initiation Protocol (SIP) Session Policies. Internet-Draft draft-ietf-sip-session-policy-framework-10, Internet Engineering Task Force. Work in progress.

[60] V. Hilt, A. Mankin, and M. Hofmann. 2004. A Framework for SIP Session Policies. Bell Labs Technical Journal 9, no. 3, pages 45–56.

[61] V. Hilt, D. Worley, G. Camarillo, and J. Rosenberg. 2011. A User Agent Profile Data Set for Media Policy. Internet-Draft draft-ietf-sipping-media-policy-dataset-12, Internet Engineering Task Force. Work in progress.

[62] ITU-T. 1998. Multipoint communication service - Service definition. Recommendation T.122, International Telecommunication Union.

[63] ITU-T. 1999. Signaling System No. 7; ISDN User Part Signaling procedures. Recommendation Q.764, International Telecommunication Union.

[64] ITU-T. 2004. Interworking between Session Initiation Protocol (SIP) and Bearer Independent Call Control Protocol or ISDN User Part. Recommendation Q.1912.5, International Telecommunication Union.

[65] ITU-T. 2007. Data Protocols for Multimedia Conferencing. Recommendation T.120, International Telecommunication Union.

[66] ITU-T. 2007. Generic Conference Control. Recommendation T.124, International Telecommunication Union.

[67] ITU-T. 2009. Packet-based Multimedia Communications Systems. Recommendation H.323, International Telecommunication Union.

[68] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne. 2011. REsource LOcation And Discovery (RELOAD) Base Protocol. Internet-Draft draft-ietf-p2psip-base-19, Internet Engineering Task Force. Work in progress.

[69] A. Jungmaier, E. Rescorla, and M. Tuexen. 2002. Transport Layer Security over Stream Control Transmission Protocol. RFC 3436.

[70] R. Kazman, M. Barbacci, M. Klein, S. Jeromy Carriere, and S. Woods. 1999. Experience with Performing Architecture Tradeoff Analysis. In: Proceedings of International Conference on Software Engineering '99 (ICSE'99), pages 54–63.

[71] J. Kempf, R. Austein, and IAB. 2004. The Rise of the Middle and the Future of End-to-End: Reflections on the Evolution of the Internet Architecture. RFC 3724.

[72] A. Keränen. 2011. Encrypted Signalling Transport Modes for the Host Identity Protocol. RFC 6261.

[73] A. Keränen, G. Camarillo, and J. Mäenpää. 2011. Host Identity Protocol-Based Overlay Networking Environment (HIP BONE)Instance Specification for REsource LOcation And Discovery (RELOAD). Internet-Draft draft-ietf-hip-reload-instance-04, Internet Engineering Task Force. Work in progress.

[74] J. Klensin. 2008. Simple Mail Transfer Protocol. RFC 5321.

[75] M. Komu, T. Henderson, H. Tschofenig, J. Melen, and A. Keranen. 2010. Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators. RFC 5770.

[76] P. Koskelainen, J. Ott, H. Schulzrinne, and X. Wu. 2006. Requirements for Floor Control Protocols. RFC 4376.

[77] P. Koskelainen, H. Schulzrinne, and X. Wu. 2002. A SIP-based Conference Control Framework. In: Proceedings of the 12th international Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), pages 53–61.

[78] J. Lennox, H. Schulzrinne, and J. Rosenberg. 2001. Common Gateway Interface for SIP. RFC 3050.

[79] S. Lindskog and A. Brunstrom. 2008. An End-to-End Security Solution for SCTP. In: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security (ARES), pages 526–531.

[80] J. Mäenpää and G. Camarillo. 2009. A Study on Maintenance Operations in a Chord-based Peer-to-Peer Session Initiation Protocol Overlay Network. In: Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 1–9.

[81] J. Mäenpää and G. Camarillo. 2010. Estimating operating conditions in a Peer-to-Peer Session Initiation Protocol overlay network. In: IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), pages 1–8.

[82] J. Mäenpää and G. Camarillo. 2011. Service Discovery Usage for REsource LOcation And Discovery (RELOAD). Internet-Draft draft-ietf-p2psip-service-discovery-03, Internet Engineering Task Force. Work in progress.

[83] J. Mäenpää, G. Camarillo, and J. Hautakorpi. 2011. A Self-tuning Distributed Hash Table (DHT) for REsource LOcation AndDiscovery (RELOAD). Internet-Draft draft-ietf-p2psip-self-tuning-04, Internet Engineering Task Force. Work in progress.

[84] R. Mahy. 2004. A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP). RFC 3842.

[85] R. Mahy, P. Matthews, and J. Rosenberg. 2010. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766.

[86] T. Moors. 2002. A Critical Review of "End-to-end Arguments in System Design". In: Proceedings of the IEEE International Conference on Communications (ICC), pages 1214–1219.

[87] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. 2008. Host Identity Protocol. RFC 5201.

[88] A. Niemi. 2004. Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903.

[89] P. Nikander, J. Arkko, T. Aura, G. Montenegro, and E. Nordmark. 2005. Mobile IP Version 6 Route Optimization Security Design Background. RFC 4225.

[90] P. Nikander, T. Aura, J. Arkko, and G. Montenegro. 2003. Mobile IP version 6 (MIPv6) Route Optimization Security Design – Extended abstract. In: Proceedings of the IEEE Semiannual Vehicular Technology Conference (VTC2003 Fall), IP Mobility Track, pages 2004–2008.

[91] P. Nikander, J. Ylitalo, and J. Wall. 2003. Integrating Security, Mobility, and Multi-Homing in a HIP Way. In: Proceedins of Network and Distributed Systems Security Symposium (NDSS), pages 87–99.

[92] M. Nordhoff. 2006. Security Evaluation of SCTP. B.Sc. Thesis. University of Duisburg-Essen, Germany.

[93] The Network Simulator ns 2. http://www.isi.edu/nsnam/ns/index.html .

[94] J. Offutt. 2002. Quality Attributes of Web Software Applications. IEEE Software 19, pages 25–32.

[95] Open Mobile Alliance. 2009. Push to Talk Over Cellular Version 2.0 - Architecture. TS, Open Mobile Alliance.

[96] Open Mobile Alliance. 2009. Push to Talk Over Cellular Version 2.0 - Control Plane Specification. TS, Open Mobile Alliance.

[97] Open Mobile Alliance. 2009. Push to Talk Over Cellular Version 2.0 - Requirements. TS, Open Mobile Alliance.

[98] J. Ott. 1997. A Multipoint Data Communication Infrastructure for Standards-based Teleconferencing Systems. Ph.D. thesis, Technischen Universität Berlin, Germany.

[99] White Paper. 2007. High Availability, Security and Peak Performance in Hosted VoIP Deployments. Aspen Networks .

[100] V. Paxson. 2001. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. ACM SIGCOMM Computer Communnication Review (CCR) 31, no. 3, pages 38–47.

[101] S. Pierrel, P. Jokela, J. Melen, and K. Slavov. 2007. A Policy system for Simultaneous Multi-Access with the Host Identity Protocol. In: Proceedings of the 1st IEEE Workshop on Autonomic Communications and Network Management (ACNM), pages 71–77.

[102] R. Price, C. Bormann, J. Christoffersson, H. Hannu, Z. Liu, and J. Rosenberg. 2003. Signaling Compression (SigComp). RFC 3320.

[103] Java Community Process. 2003. SIP Servlet API. Java Specification Request JSR 116.

[104] Java Community Process. 2004. SDP API. Java Specification Request JSR 141.

[105] Java Community Process. 2006. JAIN SIP API Specification. Java Specification Request JSR 32.

[106] Java Community Process. 2008. SIP Servlet v1.1. Java Specification Request JSR 289.

[107] Java Community Process. 2009. IMS Services API. Java Specification Request JSR 281.

[108] Java Community Process. 2009. SIP API for J2ME. Java Specification Request JSR 180.

[109] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. 2001. A Scalable Content-Addressable Network. In: Proceedings of ACM SIGCOMM, pages 161–172.

[110] E. Rescorla and IAB. 2005. Writing Protocol Models. RFC 4101.

[111] A. B. Roach. 2002. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265.

[112] A. B. Roach, B. Campbell, and J. Rosenberg. 2006. A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists. RFC 4662.

[113] J. Rosenberg. 2002. The Session Initiation Protocol (SIP) UPDATE Method. RFC 3311.

[114] J. Rosenberg. 2007. Extensible Markup Language (XML) Formats for Representing Resource Lists. RFC 4826.

[115] J. Rosenberg. 2007. The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). RFC 4825.

[116] J. Rosenberg. 2010. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245.

[117] J. Rosenberg, G. Camarillo, and D. Willis. 2006. Requirements for Consent-Based Communications in the Session Initiation Protocol (SIP). RFC 4453.

[118] J. Rosenberg, G. Camarillo, and D. Willis. 2008. A Framework for Consent-Based Communications in the Session Initiation Protocol (SIP). RFC 5360.

[119] J. Rosenberg and C. Jennings. 2008. The Session Initiation Protocol (SIP) and Spam. RFC 5039.

[120] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. 2008. Session Traversal Utilities for NAT (STUN). RFC 5389.

[121] J. Rosenberg and H. Schulzrinne. 2002. An Offer/Answer Model with Session Description Protocol (SDP). RFC 3264.

[122] J. Rosenberg and H. Schulzrinne. 2002. Reliability of Provisional Responses in Session Initiation Protocol (SIP). RFC 3262.

[123] J. Rosenberg, H. Schulzrinne, and G. Camarillo. 2005. The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP). RFC 4168.

[124] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. 2002. SIP: Session Initiation Protocol. RFC 3261.

[125] J. Rosenberg, H. Schulzrinne, and O. Levin. 2006. A Session Initiation Protocol (SIP) Event Package for Conference State. RFC 4575.

[126] A. Rowstron and P. Druschel. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329–350.

[127] J. H. Saltzer, D. P. Reed, and D. D. Clark. 1984. End-to-end Arguments in System Design. ACM Transactions on Computer Systems (TOCS) 2, no. 4, pages 277–288.

[128] M. Särelä, T. Rinta-Aho, and S. Tarkoma. 2008. RTFM: Publish/subscribe Internetworking Architecture. In: Proceedings of the ICT Mobile Summit, pages 1–8.

[129] M. Scharf and S. Kiesel. 2006. Head-of-line Blocking in TCP and SCTP: Analysis and Measurements. In: Proceedings of the Global Communications Conference (GLOBECOM), pages 1–5.

[130] H. Schulzrinne and J. Rosenberg. 1998. Signaling for Internet telephony. In: Proceedings of the 6th IEEE International Conference on Network Protocols (ICNP), pages 298–307.

[131] A. Sfairopoulou. 2008. A Cross-layer Mechanism for QoS Improvements in VoIP over Multi-rate WLAN Networks. Ph.D. thesis, Universitat Pompeu Fabra, Spain.

[132] Skype. http://www.skype.com .

[133] R. Sparks. 2003. The Session Initiation Protocol (SIP) Refer Method. RFC 3515.

[134] R. Stewart. 2007. Stream Control Transmission Protocol. RFC 4960.

[135] R. Stewart, M. Tuexen, and G. Camarillo. 2007. Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures. RFC 5062.

[136] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. 2007. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061.

[137] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. 2002. Internet Indirection Infrastructure. In: Proceedings of ACM SIGCOMM, pages 73–86.

[138] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proceedings of ACM SIGCOMM, pages 1149–160.

[139] E. Unurkhaan, E. Rathgeb, and A. Jungmaier. 2004. Secure SCTP - A Versatile Secure Transport Protocol. Telecommunication Systems 27, pages 273–296.

[140] A. Vemuri and J. Peterson. 2002. Session Initiation Protocol for Telephones (SIP-T): Context and Architectures. RFC 3372.

[141] C. Vogt. 2005. Credit-Based Authorization for Concurrent IP-Address Tests. In: IST Mobile and Wireless Communications Summit, pages 1–5.

[142] C. Vogt and J. Arkko. 2007. A Taxonomy and Analysis of Enhancements to Mobile IPv6 Route Optimization. RFC 4651.

[143] B. Zhao, J. Kubiatowicz, and A. Joseph. 2001. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical report, Berkeley, CA, USA.

Aalto-DD 129/2011

ISBN 978-952-60-4401-9
ISBN 978-952-60-4402-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
Aalto School of Electrical Engineering
Communications and Networking
www.aalto.fi

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

DOCTORAL
DISSERTATIONS